多数決に基づく公開鍵決定プロトコルによる中間者攻撃対策

多数決に基づく公開鍵決定プロトコルによる中間者攻撃対策

山森　一人[a)]・猿渡　翔一郎[b)]・相川　勝[c)]

A Public-key Decision Protocol as a Countermeasure against Man-in-the-middle Attack

Kunihito YAMAMORI, Shoichiro SARUWATARI, Masaru AIKAWA

Abstract

Most SNS (Social Networking Service) and EC (Electronic Commerce) sites request and store customers' personal information. When we exchange these information through Internet, it is recommended to use encryption mechanism such as SSL or TLS. Even if we use SSL or TLS based on Public-key cryptosystem, secret information can be stolen by Man-in-the-middle attack (MITM). Currently, Certificate Authority (CA) guarantees the legitimacy of the Public-key. However, some CAs issue false certificates, and no one can guarantee the genuineness of the CAs themselves. This paper proposes Public-key decision protocol based on majority to against MITM. Proposed protocol require no CA, and it can detect substitution of Public-key. We implement prototype of proposed protocol on virtual network and show our protocol can select correct Public-key by simulations.

Keywords: MITM, Certificate Authority, Public-key cryptography

1.　はじめに

SNS（Social Networking Service）や電子商取引（Electronic Commerce:EC）では、他人から秘匿すべき個人情報を扱うことが多い。これらの情報をインターネット上でやり取りする際は、インターネットの仕組み上パケットキャプチャなどで第三者が内容を傍受できる。そのため、SSL(Secure Socket Layer)[1)] や TLS(Transport Layer Security)[1)]を用いて暗号化通信を行うことが勧められている。もっとも、公開鍵暗号を用いた暗号通信でさえ、中間者攻撃により秘密にしたいメッセージの傍受・改ざんを行うことができる。よって中間者攻撃を対策することは重要である。現在、認証局が公開鍵の正当性を保証することで中間者攻撃対策としているが、認証局はその信頼性と維持コストに問題がある[2)]。

本論文では中間者攻撃対策として多数決に基づく公開鍵決定プロトコルを提案する。提案するプロトコルは認証局を用いずに、ネットワークに参加しているユーザの多数決により公開鍵の改ざんの検出と正しい公開鍵の決定を行う。提案するプロトコルのプロトタイプを仮想ネットワーク上で実装し、実験により正しい公開鍵を決定できることを示す。

[a)]情報システム工学科教授

[b)]情報システム工学科学部生

[c)]宮崎大学工学部教育研究支援技術センター技術職員

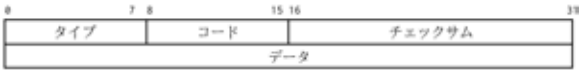

図 1　ICMP のフォーマット

2.　ICMP と暗号化通信

2.1　ICMP

ICMP(Internet Control Message Protocol)[3)]は、OSI 参照モデルのネットワーク層で動作する。このプロトコルは図 1 のようなフォーマットで、ネットワーク制御に用いられる。図 1 中のタイプとコードの意味を表 1 に示す。

ICMP を利用したコマンドの 1 つに、ping コマンドがある。ping コマンドはネットワークの疎通確認に使われ、宛先 IP アドレスに向けて Echo メッセージを送り、宛先 IP アドレスからの返事は Echo リプライメッセージで返される。ping コマンドにより、ネットワークの障害がデータリンク層以下で起きているのか、またはネットワーク層以上で起きているかを判断することができる。例えば、ウェブページをウェブブラウザで閲覧しようとして見られなかった場合、そのウェブページのドメインに対して ping コマンドを実行する。このとき ping コマンドの応答があれば、そのドメインまでのネットワークの経路は正しく接続されていると判断できるため、障害がネッ

表 1 ICMP フィールド.

タイプ	コード
0:Echo リプライメッセージ	0
3:宛先到達不可メッセージ	0:ネット到達不可能
	1:ホスト到達不可能
	2:プロトコル到達不可能
	3:ポート到達不可能
	4:フラグメントが必要だが DF はセットされている
	5:ソースルート失敗
8:Echo メッセージ	0

表 2　RSA 暗号のパラメータとその説明.

数値	説明
p	大きな素数
q	Pとは別の大きな素数
n	$n = pq$
λ	$\lambda = \gcd((p-1),(q-1))$
e	λと互いに素となる数
d	$d = e^{-1}(mod\ \lambda)$

トワーク層以上であると考えられる。ping の応答がない場合は、イーサネットケーブルの断線やルータの故障などが該当し、データリンク層以下の障害と分かる。

2.2　公開鍵暗号

公開鍵暗号は、公開鍵と秘密鍵の 2 つの鍵を用いる暗号方式である。公開鍵は通信したい相手に渡す必要があり、誰から見られてもよい。一方、秘密鍵は自分以外に知られてはならない。

公開鍵暗号の 1 つである RSA 暗号 [1,4]について簡単に説明する。RSA 暗号は 1977 年に Rivest, Shamir, Adleman によって発明された。RSA 暗号に必要な数値は p, q, n, λ, e, dであり、それぞれの意味を表 2 に示す [1]。この中で公開鍵はe, nで、秘密鍵はd, p, qとなる。メッセージmに対し、暗号化を行う場合は式(1)の計算を行う。

$$c = m^e (mod\ n). \qquad (1)$$

暗号文cの復号については式(2)の計算を行う。

$$m = c^d (mod\ n). \qquad (2)$$

RSA 暗号の安全性については、大きな数の素因数分解を速く解くアルゴリズムが見つかっておらず、nからp, qを割り出すことが困難なことで担保されている。そのため、鍵のビット長は 1024、2048 ビットが使用されることが多い [4]。

公開鍵暗号に用いられるアルゴリズムには、上記で示した RSA 以外に、ElGamal 方式、Rabin 方式、楕円曲線暗号(elliptic curve cryptosystems:ECC)などがある [2]。安全性については、ElGamal 方式は素数pの剰余類群上の離散対数問題 [1]、Rabin 方式は RSA と同じく素因数分解の困難性 [4]で、楕円曲線暗号は楕円曲線上の離散対数問題で保証している [1]。

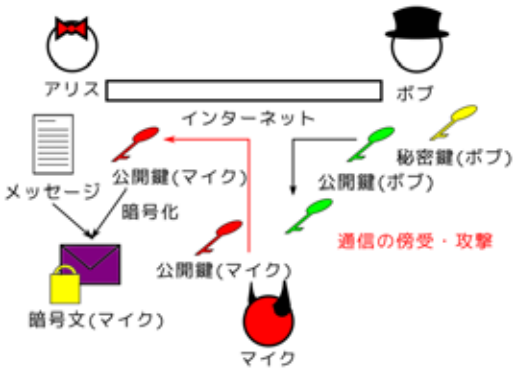

図 2　鍵入手時における中間者攻撃.

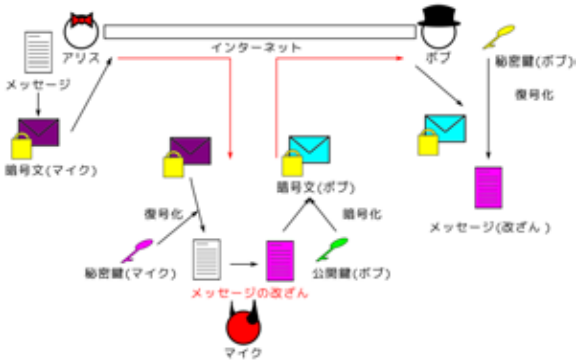

図 3　メッセージ送信時における中間者攻撃.

2.3　中間者攻撃

公開鍵暗号には公開鍵が誰のものか分からないという問題があり、この問題を利用した攻撃として中間者攻撃(Man-in-the-middle attack:MITM)がある。例として、ユーザ・アリスとユーザ・ボブの通信に対して、能動的攻撃者マイクが MITM を行う場合を図 2 と図 3 を用いて説明する。

前提として、マイクはアリスとボブのすべての通信を傍受することができる。アリスがマイクの公開鍵をボブの鍵と偽って入手させられるときの手順は図 2 の通りであり、以下にその手順を説明する。

1. アリスはボブに公開鍵（ボブ）を要求する。
2. ボブはアリスに公開鍵（ボブ）を渡そうとする。

3. マイクはボブが送った公開鍵（ボブ）を奪いとる。
4. マイクはアリスに公開鍵（マイク）を渡す。
5. アリスは入手した公開鍵（マイク）を使って秘密にしたいメッセージを暗号化する。

次に、図 3 を使って、アリスがボブに暗号文を送信するときの手順を説明する。

1. アリスは暗号文（マイク）をボブに送信しようとする。
2. マイクはアリスが送信した暗号文を奪いとる。
3. マイクは秘密鍵（マイク）を使用して暗号文を復号する。
4. マイクはメッセージの改ざんを行う。
5. マイクは公開鍵（ボブ）で改ざんしたメッセージを暗号化する。
6. マイクはボブに暗号文（ボブ）を送信する。
7. ボブは受信した暗号文を秘密鍵（ボブ）を使用して復号する。

以上のマイクの MITM により、アリスのメッセージは改ざんされうる。

2.4 認証局

MITM は、公開鍵の正当性が 1 人では判断できない点が原因で起こる。この問題を解決するために現在用いられている手法は、認証局(Certificate Authority:CA)による公開鍵の認証である。CA は公開鍵に対する署名を作成し、証明書を作成する。この署名の確認を行うことで公開鍵の所有者を確認することができる。しかし、自己署名証明書の存在や認証局自身の信頼性をどのように確保するかなどの問題は残る [2]。

3. OpenFlow

3.1 OpenFlow とは

OpenFlow は、Software-Defined Networking(SDN)における、最初の標準化されたプロトコルである [5]。従来、ネットワーク機器はデータ転送とネットワーク制御を 1 台の機器で行っていた。一方、OpenFlow ではデータ転送とネットワーク制御を、OpenFlow スイッチと OpenFlow コントローラにそれぞれ分離している。これにより、複数台の OpenFlow スイッチを 1 台の OpenFlow コントローラで制御できるなど、ネットワーク制御を柔軟に行えるようになり、ネットワークの環境変化に簡単に対応できるようになった。

OpenFlow スイッチと、OpenFlow コントローラは、暗号化通信により安全な通信路で OpenFlow プロトコルで通信を行う。OpenFlow プロトコルのメッセージをいくつか挙げると、OFPT_HELLO Message、Packet-In Message、Flow Removed Message などがある。OFPT_HELLO Message は OpenFlow コントローラと OpenFlow スイッチの接続時に使用し、Packet-In Message は OpenFlow スイッチにパケットが到着した場合に、Flow Removed Message は OpenFlow スイッチのフローエントリが削除された場合に使用されるメッセージである。

表 3 フローエントリの主構成要素.

マッチフィールド	優先度	カウンタ	インストラクション	タイムアウト	クッキー

表 4 マッチフィールド（抜粋）とその説明.

マッチフィールド名	説明
in_port	受信ポートのポート番号
eth_dst	Ethernet の宛先 MAC アドレス
eth_src	Ethernet の送信元 MAC アドレス
ip_proto	IP のプロトコル種別
ipv4_src	IPv4 の送信元 IP アドレス
tcp_src	TCP の送信元ポート番号
udp_src	UDP の送信元ポート番号

3.2 OpenFlow スイッチ

OpenFlow スイッチは表 3 のようなフローエントリを自身のフローテーブルに持つ [6]。フローエントリの各要素は以下のような意味を持つ。

マッチフィールド パケットのヘッダと一致するかどうかの条件を表し、一部抜粋したもの [7]を表 4 に示す。

優先度 0 から 65535 の値で数値が大きいほど優先度が高い。

カウンタ 届いたパケットの数を示す。

インストラクション マッチフィールドに一致した場合の動作を定義するもので、表 5 に示した種類がある [7]。

タイムアウト フローエントリの統計量やアイドル時間を示す。

クッキー フローエントリを識別するための値を示す。

3.3 OpenFlow コントローラ

OpenFlow コントローラは、OpenFlow スイッチと OpenFlow プロトコルで通信を行う。主な役割は、OpenFlow スイッチにフローエントリの追加や削除を指示することである。OpenFlow スイッチをコマンドで直接扱うのは不便なため、表 6 のように様々な言語のフレームワークとして実装されている。

3.4 OpenFlow の動作

OpenFlow の動作について図 4 を用いて説明する。

まず、OpenFlow コントローラは、OpenFlow スイッチと接続時に OpenFlow プロトコルのバージョンの整合性や必要なフローエントリの情報のやりとりを行う。

OpenFlow スイッチは OpenFlow コントローラから受け取ったフローエントリをフローテーブルに書き込む。

OpenFlow スイッチはパケットが到着したとき、パケットのヘッダ情報から表 4 のマッチフィールドの条件に合うフローエントリがないか探す。条件に合ったフローエントリが見つかった場合は、パケットを IP アドレス、MAC アドレス、もしくは表 7 に示すアクションフィールド[7]で指定した先に送信する。条件に合うフローエントリが見つからなかった場合は、そのパケットを破棄する。

表 5　インストラクションとその説明.

インストラクション	説明
Goto Table（必須）	OpenFlow1.1 以降では、複数のフローテーブルがサポートされている Goto Table によって、マッチしたパケットの処理を、指定したフローテーブルに引き継ぐことができる もし引き継ぐ場合、指定するテーブル ID は、現在のテーブル ID より大きい値でなければならない
Write Metadata（オプション）	以降のテーブルで参照できるメタデータをセットする
Write Actions（必須）	現在のアクションセットに指定されたアクションを追加する 同じタイプのアクションが既にセットされていた場合は、新しいアクションで置き換える
Apply Actions（オプション）	アクションセットは変更せず、指定されたアクションを直ちに適用する
Clear Actions（オプション）	現在のアクションセットのすべてのアクションを削除する
Meter（オプション）	指定したメータにパケットを適用する

表 6　OpenFlow コントローラとその記述言語.

コントローラ名	記述言語
Beacon	Java
Floodlight	Java
NOX	C++
POX	Python
Ryu	Python
Trema	Ruby

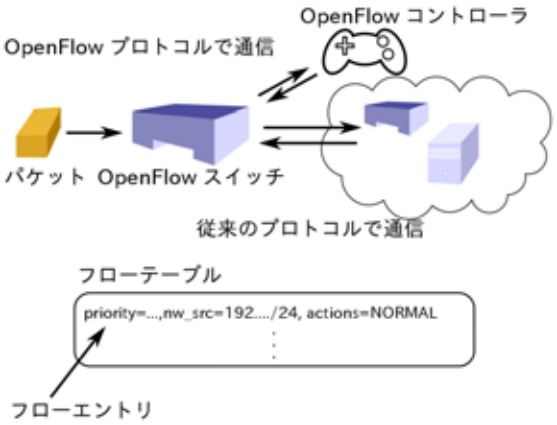

図 4　OpenFlow によるパケット処理.

4.　提案手法と評価

4.1　多数決プロトコル

提案する多数決プロトコルについて説明する。多数決プロトコルのプロトコルフォーマットを図 5 に示す。

開始コード　目印として先頭に 0 を 32 個並べたデータが格納されている。

タイプ　パケットの分類を示すもので、詳細を表 8 に示す。

コード　タイプに対するオプションで、詳細を表 8 に示す。

チェックサム　タイプとコードから算出されるチェックサムである。

要求 IP アドレス　アクションを起こす IP アドレスである。

公開鍵 IP アドレス　公開鍵を要求するときに指定する IP アドレスである。

ルータ ID　ルータを一意に定める値である。

鍵データ　公開鍵 IP アドレスの公開鍵のデータを保存する。この値は 32 ビットの整数倍でなくてもよい。

多数決プロトコルのフォーマットは ICMP をベースにしており、送信元ポート番号は 41020、宛先ポート番号は 41022 を使用する。

表 7　アクションフィールドとその説明.

アクションフィールド	説明
OFPP_IN_PORT	受信ポートに転送される
OFPP_TABLE	Packet-Out メッセージでのみ使用できるポートで、フローテーブルによって宛先ポートを決定する
OFPP_NORMAL	スイッチの L2/L3 機能で転送される
OFPP_FLOOD	受信ポートやブロックされているポートを除く当該 VLAN 内のすべての物理ポートにフラッディングされる
OFPP_ALL	受信ポートを除くすべての物理ポートに転送される
OFPP_CONTROLLER	OpenFlow コントローラに Packet-In メッセージとして送られる
OFPP_LOCAL	スイッチのローカルポートを示す
OFPP_ANY	パケット転送では使用されない

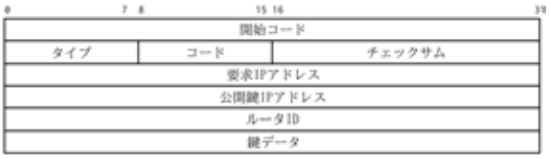

図 5　多数決プロトコルのフォーマット.

表 8　多数決プロトコルのタイプとコード.

タイプ	コード
0:鍵登録	0:登録
	1:登録済
1:プロトコル参加	0:参加要求
2:鍵要求	0:鍵要求
	1:鍵応答

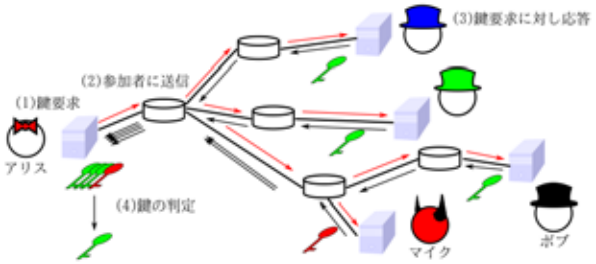

図 6　公開鍵の要求.

表 9　実行環境.

Ubuntu(OS)	Version 14.04.1
メモリ	8GB
Open vSwitch	Version 2.0.2
Ryu	Version 3.15
セグメント数	6
ルータ数	30
ルータ数あたりのホスト数	2

4.2　多数決プロトコルの動作

4.2.1　多数決プロトコルへの参加

多数決プロトコルに参加するには参加要求パケットを送信する。参加要求パケットを送信することにより、ネットワーク上のコントローラが持つ参加者リストに登録される。参加者リストは、参加要求パケットを送信した IP アドレスのリストである。

4.2.2　公開鍵の登録

公開鍵を登録するには、鍵登録パケットを送信する。鍵登録を参加要求と別にしている理由は、参加要求時に IP アドレスのなりすましにより、偽装した鍵を登録されにくくするためである。そのため、登録時には、自身の IP アドレスだけでなくルータ ID も使用する。

4.2.3　公開鍵の要求

ある公開鍵を取得したい場合、鍵要求パケットを送信する。そのときの動作を図 6 に示す。ユーザ・アリスがユーザ・ボブの公開鍵を入手する場合を例に説明する。

鍵要求　アリスはボブの公開鍵を入手するために、ボブの IP アドレスを指定した鍵要求パケットを送信する。

参加者に送信　ルータは鍵要求パケットを受信すると、受信ポート以外のポートに鍵要求パケットを送信する。

鍵要求に対し応答　ネットワーク上の多数決プロトコル参加者は、鍵要求パケットを受信したとき、自身の持つ公開鍵リストからボブの公開鍵を検索する。見つかった場合は要求 IP アドレスに対し鍵データ

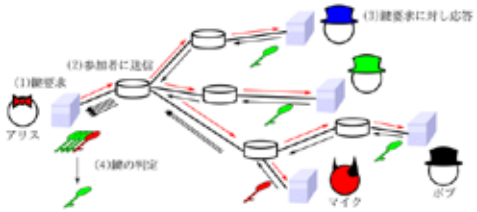

図 7　多数決プロトコルのネットワーク図.

表 10　実験環境の IP アドレッシング.

仮想の国	サブネットマスク
A	172.16.0.0/16
B	172.17.0.0/16
C	172.18.0.0/16
D	172.19.0.0/16
E	172.20.0.0/16
F	172.21.0.0/16

を付加して送信する。

鍵の判定　アリスは一定時間後、返答のあった鍵の数を種類毎に数え、多数決によりボブの公開鍵を決定する。

4.3　実験環境

実験環境を表 9 に示す。プロトコルの実装には OpenFlow を用いる。参加者リストを保存するコントローラは、OpenFlow コントローラ上に実装する。使用したコンピュータが 1 台であるため、仮想ネットワークを作成できる mininet[8]を使用した。世界中で使用されることを想定し、図 7 に示す通り、ネットワークには A 国から F 国の計 6 カ国の仮想的な国を設定する。

また、それぞれの国の IP アドレスの振り分けは表 10 の通りとし、各ルータの IP アドレスは第 3 オクテットに 1 から 7、第 4 オクテットに 1、サブネットマスクを 24 として、同一仮想国内でもセグメントを分ける。

4.4　評価と考察

提案手法で正しい公開鍵を選択できるか評価を行う。また、公開鍵をどの程度の時間で何個集めることができるかについても仮想ネットワーク上で計測を行う。

図 7 のネットワークで 60 台のホストのうち、全体で 40%のホストのボブの鍵をマイクの鍵に置き換えておく。アリスがボブの IP アドレスに対し鍵要求パケットを実際に送信する。3 秒待った後収集できた鍵の個数と正しい鍵の個数を調べる。この実験を 5 回行い平均を求める。公開鍵の選択においては、図 8 に示す通りボブを含めた 60%はボブの公開鍵を、残りの 40%はマイクの公開鍵をアリスに送信する。図 9 に示す通りアリスは返答のあった

図 8　参加者の鍵要求への応答.

図 9　多数決による公開鍵の決定.

鍵で多数決を取ることにより、正しいボブの公開鍵を選択することができている。

公開鍵の回収については図 10 に示す。鍵要求パケットを送信してから 1 秒後までは鍵は集っていないが、1 秒後から 2 秒後までで約 50 個集まっている。2 秒後から 3 秒後にかけては緩やかに鍵が集まり最終的に 55 個の鍵を回収し、正しい鍵は 33 個だった。このため、鍵の収集には 2 秒程度必要だと考えられる。

5.　おわりに

本論文では、公開鍵認証の問題である中間者攻撃を認証局を用いずに防ぐことを目的とし、多数決により公開鍵を決定するプロトコルを提案した。このプロトコルはネットワークに参加しているユーザの多数決により正しい公開鍵を決定する。実際にプロトコルのプロトタイプを仮想ネットワーク上で実装し、正しい公開鍵を決定できることを示した。鍵要求を行う実験用プログラムにおいて、鍵要求パケットの応答に対する返答の時間経過を調べたところ、2 秒以上必要であることが分かった。しかし、実ネットワークでも同様の結果になるかは確認する必要がある。

今後の課題は、公開鍵を IP アドレスとは別の識別子を用いて対応付けを行うこと、全ユーザが参加者全員の公開鍵を持つのではなく、地域ごとなど特定の公開鍵のみを持つようにすること、鍵要求時の参加者リストからルータ ID などを用いて特定のユーザを選択すること、攻撃者を排除する仕組みを用意することなどが挙げられる。

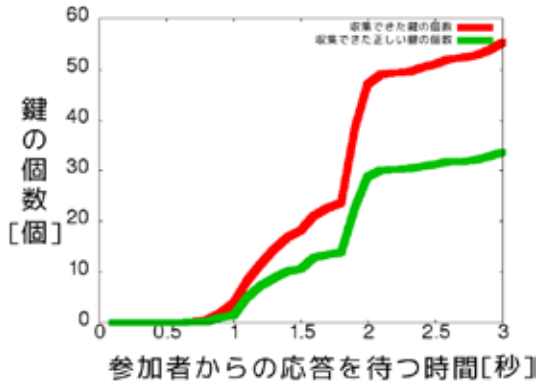

図 10　鍵要求パケットに対する返答の時間経過の様子.

参考文献

1) 佐々木　良一，吉浦　裕，手塚　悟，三島　久典: インターネット時代の情報セキュリティ　-暗号と電子透かし-. 共立出版株式会社，October 2000. 初版 1 刷
2) 結城　浩:暗号技術入門　-秘密の国のアリス.SB クリエイティブ株式会社，新版，January 2014. 初版第 9 刷
3) J.Postel: Internet Control Message Protocol.https://www.ietf.org/rfc/rfc792.txt, September 1981.
4) Dougras R. Stinson, 櫻井　幸一監訳: 暗号理論の基礎. 共立出版株式会社，March 1998. 初版 3 刷.
5) Open Networking Foundation: ONF Overview – Open Networking Foundation. https://www.opennetworking.org/ja/about-ja/onf-overview-ja,January,2015.
6) Open Networking Foundation: OpenFlow Switch Specification. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf, June 2012.
7) RYU project team: RYU SDN Framework. http://osrg.github.io/ryu-book/ja/Ryubook.pdf, September 2014.
8) Mininet Team: Minint An instant virtual network on your laptop (or other PC) – mininet. http://mininet.org, January 2015.