情報処理学会論文誌 Vol.60 No.3 830–838 (Mar. 2019)

不正パケットの高速な検出を実現する簡易認証方式の提案と評価

鴨下 友馬[1,a)] 鈴木 秀和[1,b)] 内藤 克浩[2,c)] 渡邊 晃[1,d)]

受付日 2018年6月4日, 採録日 2018年12月4日

概要：暗号通信においてはDoS攻撃やリプレイ攻撃に対する対策が重要である．暗号通信では事前に送信側と受信側が共通鍵を共有しているため，HMAC（Hash-based Message Authentication Code）を用いたパケット検証を行い，不正パケットを廃棄するという処理が一般である．しかし，HMACはパケットサイズが大きいと検証に時間を要するという課題がある．そこで，本論文では共通鍵を用いてシーケンス番号のみを最初にチェックする簡易認証方式を提案する．簡易認証を従来のパケット検証処理の事前処理として実行することにより，不正パケットを高速に検出して廃棄できる．簡易ハッシュの計算時間は短いので，正常なパケットの処理に与える影響は小さい．実験とシミュレーションにより提案方式の評価を行い，有用性が高いことを示した．

キーワード：DoS攻撃，リプレイ攻撃，暗号通信，パケット認証

Proposal and Evaluation of Simple Authentication Method that Detects Invalid Packets Fast

Yuma Kamoshita[1,a)] Hidekazu Suzuki[1,b)] Katsuhiro Naito[2,c)] Akira Watanabe[1,d)]

Received: June 4, 2018, Accepted: December 4, 2018

Abstract: For cipher communication, countermeasures against DoS attacks and replay attacks are essential. In cipher communication, since the initiator and the responder share a common key beforehand, a process of packet verification using Hash-based Message Authentication Code (HMAC) is common, and illegal packets are discarded in this process. However, HMAC has a problem that it takes time to verify if the packet size is large. Therefore, we propose a simple authentication method that first checks only the sequence in a packet using the common key. By executing simple authentication before the conventional packet authentication, illegal packets can be discarded fast. Since the calculation time of the simple hash is short, the influence on the processing of the normal packet is small. We evaluated by experiment and simulation and showed that the proposed method is useful.

Keywords: DoS attack, replay attack, cipher communication, packet verification

1. はじめに

携帯電話やスマートフォン等の移動通信端末の普及や，IoT（Internet of Things）の発展にともない，ネットワークを利用する機会がますます増加している．このような状況において，ネットワークセキュリティを実現する暗号通信はきわめて重要な技術である．暗号通信では，通信開始時のシグナリング処理で相手認証と共通鍵の共有が行われ，以後のデータ通信では共通鍵による高速暗号通信を行

[1] 名城大学理工学部
Faculty of Science and Technology, Meijo University, Nagoya, Aichi 468–8501, Japan
[2] 愛知工業大学情報科学部
Faculty of Information Science, Aichi Institute of Technology University, Toyota, Aichi 470–0392, Japan
[a)] yuma.kamoshita@wata-lab.meijo-u.ac.jp
[b)] hsuzuki@meijo-u.ac.jp
[c)] naito@pluslab.org
[d)] wtnbakr@meijo-u.ac.jp

うのが一般的な手法である．相互認証と暗号化により第三者が通信の内容に関与できないという特徴がある．

しかし，暗号通信に対しても様々なサイバー攻撃が存在する．なかでも，DoS 攻撃（Denial of Service Attack）とリプレイ攻撃（Replay Attack）は大きな脅威である [1]．これらの攻撃に対しては，不正パケットを可能な限り早く検出し，廃棄することが重要である．

DoS 攻撃は，大量のデータをターゲットに送りつけ，ターゲットの動作を妨害するが，攻撃パケットと正常なパケットの区別ができないので対処が難しい攻撃である．一般的に DoS 攻撃は，攻撃者を特定できないように，送信元を偽造する場合が多い．そのため DoS 攻撃対策の例としては，シグナリング処理の開始時に，軽いやりとりを行うことにより，通信相手のアドレスが確かに存在することを確認してから認証等の重い処理に移行するという対処方法をとる．その後のデータ通信では，共有した共通鍵を利用してパケットの MAC（Message Authentication Code）検証を行い，不正パケットを確実に廃棄する方法が一般である．

リプレイ攻撃は正常なパケットをモニタして再利用する攻撃で，MAC チェックを通り抜ける可能性がある．そのため，MAC 認証とは別に専用の対策をとる必要がある．一般にはシーケンス番号を用いた不正パケットの判別が行われる．

既存の暗号通信におけるサイバー攻撃対策を IPsec（Security Architecture for the Internet Protocol）[2] を例にとり説明する．IPsec では，不正パケットの検出を以下のように行うよう定められている．最初に，パケット内に付与されたシーケンス番号を用いてリプレイ攻撃チェックを行う．受信側はリプレイ防御ウィンドウと呼ばれるビットマスクを用いて受信を許可するシーケンス番号の範囲を決定する．範囲を逸脱したパケットはリプレイ攻撃と判断し廃棄する．

続いて，DoS 攻撃チェックとして，HMAC（Hash-based Message Authentication Code）[3] による MAC 認証を行う．MAC 認証が不成功の場合，通信相手が共通鍵を保持していないと判断し，パケットを廃棄する．MAC 認証に成功した場合，リプレイ対策用のウィンドウを更新する．以上により，1 パケットの処理が完了し，正常な受信処理にパケットが渡される．

この方式は，不正パケットの検出にかかる多くの時間が MAC 認証処理に依存する．MAC 認証はすべての内容を検証するため，パケット長が長いと処理時間が長くなるという課題がある．

そこで，本論文ではパケット内に短いチェック用フィールドを新たに定義する簡易認証方式を提案する．提案方式では，送信側は共通鍵とシーケンス番号から簡易ハッシュ値を生成してパケット内に付加する．受信側はこの付加情報を最初に検証することにより，ほとんどの不正パケットを短時間で廃棄することができる．簡易認証では不正パケットを完璧にチェックすることはできないが，この場合は正規の MAC 認証で最終的に必ず廃棄される．簡易認証は処理が軽いので，正規の受信処理にほとんど影響を与えることはない．ただし，提案方式を適用するための条件として，パケット内に簡易ハッシュ値を格納するフィールドが必要である．このフィールドは改ざんされていないことが保証された平文領域でなければならない．

本論文では，提案方式を用いたパケット検証の実験を行い，パケット検証に要する処理時間を計測した結果を示す．実験には上記条件を満たす NTMobile [4], [5], [6], [7], [8], [9] を使用した．また，実測値をもとにシミュレーションを行い，パケット検証に要する処理時間の総合的な評価を行った結果，提案方式が有用であることを示す．さらに簡易ハッシュ値のサイズを評価し，8 bit のフィールドがあれば十分であることを示す．

以降，2 章では既存技術によるパケット検証について説明する．3 章では提案方式を用いたパケット検証について説明し，4 章では提案方式を実装したテストプログラムの概要について示す．5 章ではテストプログラムの動作検証および処理時間の測定，シミュレーションから評価を行い，最後に 6 章でまとめる．

2. 既存技術

本章では，IPsec と，NTMobile（Network Traversal with Mobility）のデータパケットについて，パケット検証処理をどのように実現しているかを説明する．IPsec にはパケット認証のみを行う AH（Authentication Header）[10] と，暗号化とパケット認証を行う ESP（Encapsulating Security Payload）[11] がある．本論文では暗号通信を対象とすることから，ESP を取り上げて説明する．

2.1 ESP の検証処理

2.1.1 ESP のフォーマット

ESP は，パケットの機密性[*1]および完全性[*2]を保証するセキュリティプロトコルである．通信開始時に IKE（Internet Key Exchange）[12] により相互認証と共通鍵の共有を行っておく必要がある．機密性の確保には共通暗号鍵によるパケットの暗号化，完全性の確保には同様に共通認証鍵による ICV（Integrity Check Value）による完全性チェックを行う．ICV はその役割という観点では MAC（Message Authentication Code）と同義である．したがって，以下では ICV をチェックする処理を「MAC 認証」と呼ぶ．

図 1 に，ESP トランスポートモードのパケットフォー

[*1] アクセスを認可された者だけが情報にアクセスできることを確実にする性質．
[*2] 情報および処理方法が正確であること，および完全であることを保護する性質．

図 1 ESP のパケットフォーマット
Fig. 1 Packet format of ESP.

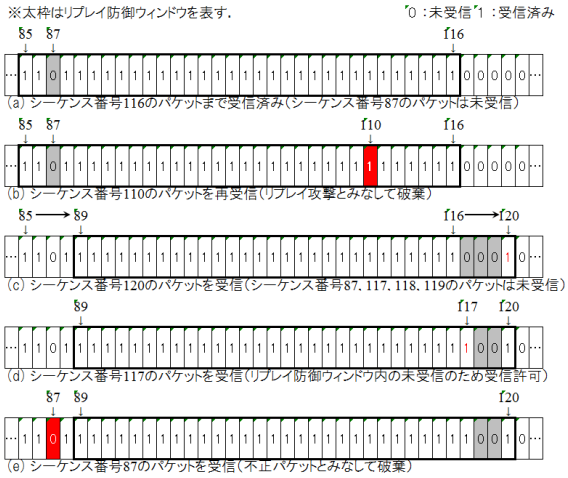

図 2 リプレイ攻撃チェックの概要
Fig. 2 Overview of the checking process of replay attack.

マットを示す．encrypted は暗号化範囲，authenticated は認証範囲であり，後者が MAC 認証の対象となる．ESP header には 32 bit のシーケンス番号が含まれている．ICV は 128 bit であり，パケット認証範囲のデータと共通認証鍵から HMAC-MD5 により生成し，その結果を ICV フィールドに付与する．

2.1.2 ESP のパケット検証処理

パケット検証は，リプレイ攻撃チェック，MAC 認証の順に行う．不正パケットであると判定した場合にはその時点で廃棄を決定し，以降の処理は行わない．MAC 認証まで成功した場合は正規のパケットと見なし，リプレイ防御ウィンドウの更新（受信許可範囲の変更処理）を行う．最後に，パケットを復号して受信処理に入る．

リプレイ攻撃チェックでは，リプレイ防御ウィンドウと呼ばれる 32 bit 以上のビットマスクを用いて受信を許可するシーケンス番号の範囲を決定し，リプレイ攻撃パケットを検出する．リプレイ攻撃は正規のパケットをモニタして再利用するため，MAC 認証では検出できず，必須の処理である．

図 2 に，リプレイ攻撃チェックの概要を示す．ここで，リプレイ防御ウィンドウのサイズは 32 bit である．シーケンス番号が 116 のパケットまで受信済みであるとすると（図 2 (a)），リプレイ防御ウィンドウ内の未受信のパケット（シーケンス番号 87），およびシーケンス番号 117 以降のパケットは受信を許可する．受信済みのシーケンス番号 110 のパケットを再度受信した場合は不正パケットと判定される（図 2 (b)）．また，シーケンス番号 120 のパケットを受信した場合は受信許可範囲が変わる（図 2 (c)）．このとき，未受信のシーケンス番号 117 のパケットは受信可能（図 2 (d)）だが，シーケンス番号 87 のパケットはリプレイ防御ウィンドウから外れるため不正パケット扱いとなる（図 2 (e)）．これは，未受信のパケットとはいえ，最新の受信済みシーケンス番号 120 と比較して古すぎるシーケンス番号のパケットをこの段階で受信するのは不自然であり，不正パケットと考えるべきだからである．

リプレイ攻撃チェックの段階では，受信許可するか破棄するかだけを決定し，ウィンドウの更新処理は行わない．これは，この時点では正規のパケットであるか否かが確定していないためである．

リプレイ攻撃チェックを通過した受信パケットは，続けて MAC 認証を行う．MAC 認証では，受信パケットの認証範囲と共通鍵を入力として，ハッシュ関数 HMAC-MD5 により 128 bit の認証コードを生成する．これと受信パケットの ICV を比較し，一致していれば受信を許可する．

MAC 認証を通過した受信パケットは，正規のパケットと見なしてリプレイ防御ウィンドウ更新処理を行う．具体的には，図 2 の (c) や (d) のように受信を許可した場合に，そのパケットを受信したことを表すフラグを立て，ウィンドウをスライドさせる処理を行う．

2.2 NTMobile の検証処理

2.2.1 NTMobile の動作

NTMobile は，通信中にネットワークが切り替わっても通信を継続できる移動透過性と，ネットワーク環境にかかわらず通信を開始することができる通信接続性の両者を同時に実現する技術である．NTMobile には通信開始時のシグナリング処理とデータの暗号通信の両者の定義が含まれている．通信開始時には，エンド端末とインターネット上に設置された NTMobile サポート装置群との間でシグナリング処理を行い，エンド端末間のトンネル経路を構築する．このシグナリング処理の中でエンド装置間が共通鍵を共有する．通信パケットは，このトンネル経路を経由してエンドツーエンドの暗号通信を行う．NTMobile のデータパケットも ESP と同様に，パケットの機密性および完全性を保証する．

2.2.2 NTMobile のフォーマットとパケット検証処理

図 3 に NTMobile のデータパケットのフォーマットを示す．また実験データに係わることから NTM header の詳細をあわせて示す．文献 [4], [5] では NTMobile のパケットフォーマットがすでに提示されているが，その後 NTM header の内容が変わっているため，あらためて現在のフォーマットを提示した．encrypted は暗号化範囲，authenticated は認証範囲であり，後者が MAC 認証の対象となる．NTM header には NTMobile の通信に関わるデータが含まれており，32 bit のシーケンス番号もここに含ま

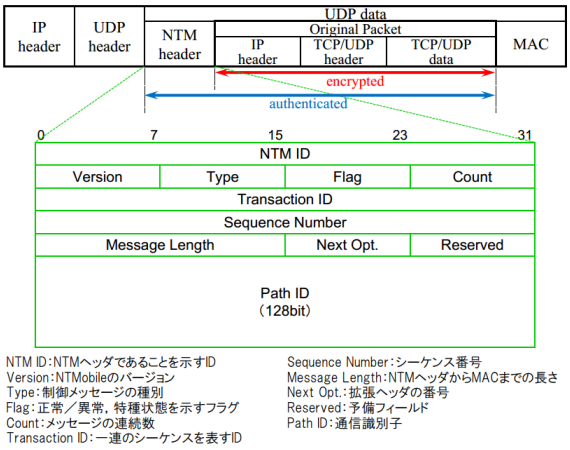

図 3 NTMobile のパケットフォーマット
Fig. 3 Packet format of NTMobile.

れる．MAC は 128 bit であり，パケットの認証範囲と共通鍵を用いて HMAC-MD5 により生成する．

パケット検証の方法は基本的に ESP と同様で，リプレイ攻撃チェック，MAC 認証の順に行う．不正パケットであると判定した場合にはその時点でパケットを廃棄し，以後の処理は行わない．MAC 認証まで成功した場合は正規のパケットと見なし，リプレイ防御ウィンドウの更新（受信許可範囲の変更処理）を行う．検証に成功したパケットは復号して受信処理に渡される．

2.3 既存方式の課題

パケット検証処理の中で最も時間を要するのは MAC 認証処理である．MAC 認証は原理上パケット長が長いほど処理時間が長くなるという課題がある．不正パケットは早期に発見して廃棄することが DoS 攻撃耐性を向上させるうえで重要である．特に NTMobile には，RS（Relay Server）と呼ばれるパケット中継装置が存在する．RS では大量のパケットを中継するため，不正パケットを可能な限り短時間で検出する機能が求められる．

3. 提案方式

本章では，提案する簡易認証の概要，および簡易認証に用いるハッシュ関数について説明する．

3.1 簡易認証におけるパケット検証処理

簡易認証は，不正パケットを高速に検出するためパケットの一部の情報，具体的にはシーケンス番号だけを高速に検証する．簡易認証は処理に要する時間が MAC 認証と比較して短いため，パケット検証の最初に行うことで従来方式の補助的役割を担うことができる．提案方式を適用する際は，簡易認証に係るフィールドをパケット内に追加する必要がある．フィールドを追加する場所は，暗号化範囲外かつ認証範囲内である必要がある．

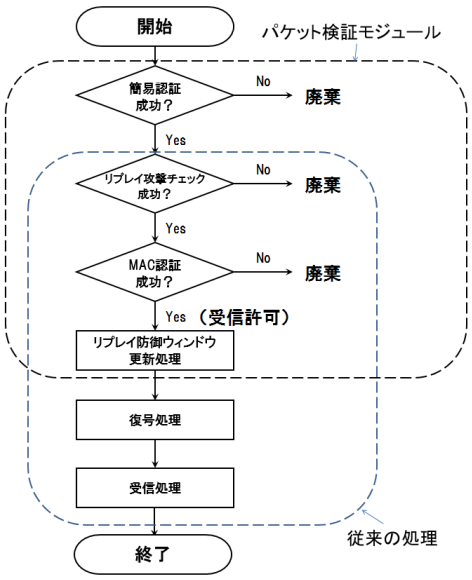

図 4 受信側におけるパケット検証処理
Fig. 4 packet checking process of the receiving side.

送信側は，通信に用いる共通鍵とシーケンス番号を入力としてハッシュ値（以下，簡易ハッシュ値）を生成し，パケットに付与する．受信側では最初に簡易ハッシュ値をチェックし，その後の処理は従来と同様とする．図 4 に受信側におけるパケット検証処理を示す．従来の処理の前に簡易認証を行う．送信側と同様の方法で簡易ハッシュ値を計算し，受信パケットに付与された簡易ハッシュ値と比較する．値が不一致であれば不正パケットであると判定して廃棄し，一致していれば既存のリプレイ攻撃チェックに進む．以後の処理は既存のパケット検証処理と同様に，MAC 認証を行い，正規のパケットであればリプレイ防御ウィンドウの更新を行う．パケット検証に関わる処理が終わると，その後は復号処理を経て正規の受信処理を行う．

3.2 簡易認証に用いるハッシュ関数

簡易認証に用いるハッシュ関数は，演算時間の短い FNV-1（32 bit）を使用することとした [13]．FNV-1（32 bit）の演算方法は以下のとおりである．入力を $\boldsymbol{M} = \{m_1, m_2, \cdots, m_n\}$ とすると，出力 $h(\boldsymbol{M})$ は (1) のようになる．

$$h(\boldsymbol{M}) = \{\{\cdots\{\{((Of \times Pr) \oplus m_1) \times Pr\} \oplus m_2\} \cdots\} \times Pr\} \oplus m_n \quad (1)$$

ここで，Of（Offset），Pr（Prime）は定数であり，それぞれ $Of = 2{,}166{,}136{,}261$，$Pr = 16{,}777{,}619$ である．FNV-1（32 bit）には，以下のような特徴がある．

性質 1 入力 $\boldsymbol{M}$ は 8 bit 整数のベクトルである．すなわち，$m_1, m_2, \cdots, m_n$ はいずれも 8 bit 整数である．

性質 2 入力 $\boldsymbol{M}$ において m_n が近接した値のとき，$h(\boldsymbol{M})$

の値が近くなることがある．

性質 3 出力は 32 bit 整数である．

性質 2 については，4.2 節で説明する独自の工夫により値が分散するようにした．

4. 実装

本章では，提案方式を実装して実験した試作システムの概要について示す．

4.1 パケット検証モジュール

パケット検証モジュールは，不正パケットを検出し廃棄するモジュールである．図 4 に示す検証処理のうち，「簡易認証」，「リプレイチェック」，「MAC 認証」，「リプレイ防御ウィンドウ更新処理」の処理を行う．NTMobile に適用する場合を想定し，C 言語により記述した．図 3 の NTM header 以降の部分はダミーデータで仮生成した．NTM header には 8 bit の予備フィールドがあるため，ここに簡易認証データを格納した．

4.2 簡易ハッシュ値の生成

ハッシュ関数 FNV-1（32 bit）の入力は，共通鍵とシーケンス番号である．NTMobile の共通鍵は 128 bit 整数であり，シーケンス番号は 32 bit 整数である．したがって，入力値の合計は 160 bit であり，FNV-1（32 bit）の性質 1 より $\boldsymbol{M} = \{m_1, m_2, \cdots, m_{20}\}$ となる．

ここで，共通鍵を $\boldsymbol{K} = \{k_1, k_2, \cdots, k_{16}\}$，シーケンス番号を $\boldsymbol{N} = \{n_1, n_2, n_3, n_4\}$ と表す（$k_i\, n_j$ はいずれも 8 bit 整数）．セッション継続中は $\boldsymbol{K}$ は変化せず，$\boldsymbol{N}$ はパケットを送信するたびに 1 ずつ増加する．このとき，m_n に n_4 を配置すると，FNV-1（32 bit）の性質 2 より，隣接するシーケンス番号のパケットの $h(\boldsymbol{M})$ は近接することがある．そのため，シーケンス番号の増分から $h(\boldsymbol{M})$ の予測が容易となる可能性がある．そこで，$\boldsymbol{M} = \{\boldsymbol{N}, \boldsymbol{K}\} = \{n_1, \cdots, n_4, k_1, \cdots, k_{16}\}$ とした．

$h(\boldsymbol{M})$ は FNV-1（32 bit）の性質 3 より 32 bit となるが，NTM header に格納するため，$h(\boldsymbol{M})$ の下位 8 bit を簡易ハッシュ値として使用した．

4.3 簡易認証の手順

受信側は，受信パケットの NTM header からシーケンス番号を取得し，これと共通鍵から簡易ハッシュ値を生成する．生成した簡易ハッシュ値と，受信パケットの NTM header に付与された簡易ハッシュフィールドの値を比較し，一致していなければ不正パケットと判定してパケットを廃棄する．一致していればリプレイ攻撃チェックに進む．

4.4 リプレイ攻撃チェックの手順

NTMobile では，リプレイ攻撃チェックは現時点において未実装であるため，リプレイ攻撃チェックの実装を新たに行った．リプレイ攻撃チェックは，RFC2401（IPsec の旧バージョンに関する RFC）にサンプルプログラムが掲載されている [14]．ただし，「リプレイ攻撃チェック」と「リプレイ防御ウィンドウ更新処理」が一体となっているため，これらの処理を分割する必要がある．シーケンス番号がウィンドウの範囲外であった場合リプレイ攻撃と見なしパケットを廃棄する．範囲内であった場合は MAC 認証に進む．リプレイ防御ウィンドウの更新処理はこの時点では実行せず，次に示す MAC 認証に成功した後に実行する．

表 1 実験環境の仕様
Table 1 Specification of the experimental system.

	ホストマシン	仮想マシン
OS	Windows 7 64 bit	Ubuntu 14.04 32 bit
Linux カーネル	–	3.13.0-116-generic
CPU	i7-2600 3.40 GHz	1 Core 割り当て
Memory	8.00 GB	1.00 GB 割り当て

4.5 MAC 認証の手順

MAC 認証処理は現状の NTMobile に実装済みであり，テストプログラムではこれをそのまま引用した．HMAC-MD5 により受信パケットから MAC 値を生成し，受信パケットの MAC フィールドの値と比較する．一致していなければ MAC 認証に失敗したものとし，不正パケットと判定して廃棄する．一致していれば MAC 認証に成功したものとする．最後にリプレイ防御ウィンドウを更新し，パケット検証処理を終了する．

5. 評価

本章では，提案方式を実装したテストプログラムの動作検証および処理時間を測定した結果を示す．さらに，測定結果を利用してシミュレーションを行い提案方式の有用性評価を行った．

5.1 実験環境

表 1 に実験環境の仕様を示す．Windows マシンに NTMobile が動作する仮想の Linux マシンを立て，この中で実験プログラムを実行させた．NTM header の長さを 36 Byte，Original Packet の長さを 1,000 Byte とし，Original Packet の内容はダミーデータで仮生成した．パケット検証処理を行い，正常に動作することを確認した．

5.2 パケット検証処理の測定

パケット検証処理の内訳を測定した．**表 2** にパケット検証処理を 100,000 回実行した際の，パケット検証処理時間の平均を示す．ここで，t_s は簡易認証に要する時間，t_r はリプレイ攻撃チェックに要する時間，t_m は MAC 認証に要

表 2 パケット検証処理時間（μs）
Table 2 Processing time of packet checking process (μs).

t_s	t_r	t_m	t_u	Total
0.536	0.414	3.835	0.561	5.346

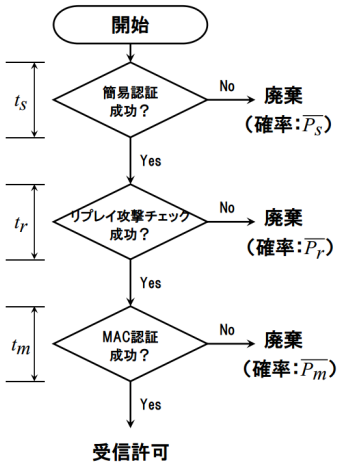

図 5 不正パケット検出処理とその確率
Fig. 5 Detection process of invalid packets and its provability.

する時間，t_u はリプレイ防御ウィンドウ更新処理に要する時間である．表 2 の結果より，MAC 認証処理 t_m がほとんどの時間を占めていることが分かる．簡易認証処理の時間 t_s は Total との比率として約 10%を占めることが分かる．

5.3 不正パケットの検証処理時間の評価

各処理において不正パケットが検出される確率が異なるので，このままでは正しく有用性を評価できない．そこで，確率を加味し，数値計算を行うことにより正確な評価を試みた．実測したそれぞれの検証処理時間と，不正パケットが検出される確率から，最終的な不正パケット検証処理時間を算出した．**図 5** に，不正パケット検出処理とその確率を示す．ある不正パケットが簡易認証で破棄される確率を $\overline{P_s}$，リプレイ攻撃チェックで破棄される確率を $\overline{P_r}$，MAC 認証で破棄される確率を $\overline{P_m}$ とすると，

$$\overline{P_s} + \overline{P_r} + \overline{P_m} = 1 \tag{2}$$

が成り立つ．このとき，不正パケットの検証処理時間の平均値 E は式 (3) のようになる．

$$E = t_s\,\overline{P_s} + (t_s + t_r)\,\overline{P_r} + (t_s + t_r + t_m)\,\overline{P_m} \tag{3}$$

次に不正パケットが簡易認証に成功する確率 P_s を算出する．前提として，不正パケットの簡易ハッシュ値の分布は一様であると仮定する．簡易ハッシュ値の長さを l_h[bit] とすると，

$$P_s = \frac{1}{2^{l_h}} \tag{4}$$

となる．よって，ある不正パケットが簡易認証で廃棄される確率 $\overline{P_s}$ は，

$$\overline{P_s} = 1 - P_s \tag{5}$$

となる．

次に，ある不正パケットがリプレイ攻撃チェックに成功する確率 P_r を算出する．リプレイ攻撃チェックにおいて受信許可と判定されるシーケンス番号は，以下のいずれかの条件を満たす必要がある．

条件 1 最新のシーケンス番号より大きく，シーケンス番号の最大値以下であるもの

条件 2 リプレイ防御ウィンドウの範囲内のうち，未受信のシーケンス番号であるもの

ここで，シーケンス番号の長さを l_n[bit] とすると，シーケンス番号の最大値は $2^{l_n} - 1$ となる．したがって，検証処理開始時点での最新の受信済みシーケンス番号を n_l（図 2 で (c) から (d) に遷移する時点での n_l は 120）とすると，条件 1 を満たすシーケンス番号の数は，

$$\left(2^{l_n} - 1\right) - n_l \tag{6}$$

となる．一方，リプレイ防御ウィンドウのサイズを s_w，検証処理開始時点でのリプレイ防御ウィンドウ内の受信済みシーケンス番号の個数を r（$1 \le r \le \min(s_w, n_l)$）（図 2 で (c) から (d) に遷移する時点での r は 29）とすると，条件 2 を満たすシーケンス番号の数は，

$$\min(s_w, n_l) - r \tag{7}$$

となる．受信許可と判定されるシーケンス番号の数は式 (6) と (7) の和であり，シーケンス番号の総数は 2^{l_n} であるから，P_r は，

$$P_r = \frac{\left\{\left(2^{l_n} - 1\right) - n_l\right\} + \left\{\min(s_w, n_l) - r\right\}}{2^{l_n}} \tag{8}$$

となる．

ある不正パケットがリプレイ攻撃チェックで廃棄されるとき，その不正パケットは簡易認証に成功している必要がある．したがって，ある不正パケットがリプレイ攻撃チェックで廃棄される確率 $\overline{P_r}$ は，

$$\overline{P_r} = P_s\,(1 - P_r) \tag{9}$$

となる．

以上から，ある不正パケットが MAC 認証で廃棄されるとき，その不正パケットは簡易認証とリプレイ攻撃チェックの両方に成功している必要がある．したがって，ある不正パケットが MAC 認証で廃棄される確率 $\overline{P_m}$ は，

$$\overline{P_m} = P_s\,P_r \tag{10}$$

となる．なお，式 (5)，(9)，(10) は，式 (2) の条件を満たしている．

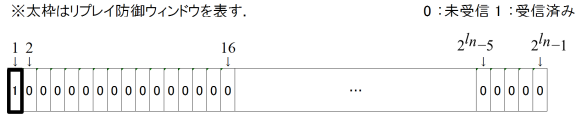

図 6 $n_l = 1$，$r = 1$ のときのリプレイ防御ウィンドウ
Fig. 6 Status of replay defence window when $n_l = 1$, $r = 1$.

簡易ハッシュ値の長さ l_h，シーケンス番号の長さ l_n，リプレイ防御ウィンドウのサイズ s_w は定数である．したがって，$\overline{P_s}$ は定数である．一方，検証処理開始時点での最新の受信済みシーケンス番号 n_l，リプレイ防御ウィンドウ内の受信済みシーケンス番号の個数 r は，受信状況により変化する値である．したがって，$\overline{P_r}$，$\overline{P_m}$ は変数である．なお，式 (9)，(10) から，$\overline{P_r}$，$\overline{P_m}$ は P_r に依存し，$\overline{P_r}$ が小さいほど $\overline{P_m}$ は大きくなることが分かる．

表 2 の実測値を得た際の各種パラメータは，$l_h = 8$，$l_n = 32$，$s_w = 32$ である．また，n_l，r は受信状況により変化するため厳密に定義することはできないが，以下では $n_l = 1$，$r = 1$ としてシミュレーションを行う．なお，$n_l = 1$，$r = 1$ は「シーケンス番号 1 のパケットのみを受信した状態」であり，リプレイ防御ウィンドウは**図 6** に示した状態となっている．このとき，リプレイ攻撃チェックで廃棄される確率 $\overline{P_r}$ は最小となるため，MAC 認証で廃棄される確率 $\overline{P_m}$ が最大となる．

以上のパラメータと表 2 の実測値 t_s，t_r，t_m により，不正パケットの検証処理時間の平均値 E を求めると，

$$E = 0.553\,[\mu\mathrm{s}] \tag{11}$$

となった．

一方，簡易認証を用いていない既存のパケット検証処理は，$l_h = 0$，$t_s = 0$ のときと同等である．その他のパラメータと実測値は同じものを用いると，不正パケットの検証処理時間の平均値 E は，

$$E|_{l_h=0,\,t_s=0} = 4.249\,[\mu\mathrm{s}] \tag{12}$$

となった．この結果から，提案手法により不正パケットの検証処理時間は (11) と (12) から 87%削減されたことが分かる．したがって，不正パケットによる DoS 攻撃耐性が大きく向上することが期待できる．

5.4 正規のパケットの検証処理時間の評価

不正パケットの検出性能が向上した一方で，攻撃がない状態においては，パケット検証につねに簡易認証処理が加わるため負荷が増えることになる．そこで，増加した負荷が全体の検証処理時間に与える影響を調査した．

簡易認証を用いていない既存のパケット検証処理における，正規のパケットの検証処理時間の実測値は，

$$t_r + t_m + t_u = 4.810\,[\mu\mathrm{s}] \tag{13}$$

表 3 MAC 認証と復号の処理時間
Table 3 Processing time of MAC authentication and decryption.

	処理時間 [μs]
MAC 認証処理	280
復号処理	2,930

表 4 処理時間測定環境の仕様
Table 4 Specification of the measurement system.

	ホストマシン	仮想マシン
OS	Windows 7 64 bit	Ubuntu 14.04 32 bit
Linux カーネル	–	3.13.0-116-generic
CPU	i7-2660 3.40 GHz	1 Core 割り当て
Memory	8.00 GB	2.00 GB 割り当て

となる．これに対して，提案手法における，正規のパケットの検証処理時間の実測値は，

$$t_s + t_r + t_m + t_u = 5.346\,[\mu\mathrm{s}] \tag{14}$$

となる．提案手法では t_s が加わるため，検証処理にかかる時間は約 11%長くなることが分かる．しかし，この後の復号処理時間，さらには正規の受信処理時間を考慮すると，t_s の影響は小さい．

表 3 に別環境にて測定した MAC 認証処理と復号処理の実測結果を示す．測定環境の仕様は**表 4** のとおりである．ホストマシン内に NTMobile 端末 2 台と，NTMobile のシグナリングをサポートする装置群 3 台が仮想マシンとして存在しており，実際に 1 KByte 長の UDP データを送受信させて測定した．測定結果は 1,000 回の平均値である．測定環境は異なるが，両者の処理時間比率はそのまま適用できる．この結果によると，復号処理時間は MAC 認証の 10.46 倍の時間を要している．復号処理時間は MAC 認証と同様にメッセージ長とともに長くなるため，MAC 認証時間と復号処理時間の比率はメッセージ長にかかわらずほぼ同じである．従来の不正パケットの検出にかかる時間を 1 とすると，復号を含む正常な処理にかかる時間は 11.46 (1+10.46)，復号を含む提案方式の正常な処理時間は 11.57 (1.11+10.46) となることから，処理時間は 0.96%増加し，スループットに換算すると，0.95%の低下と見積もることができる．正常なパケットにおいては，その後にパケットの内容を解析して分岐する処理が続くので，簡易認証によるスループット低下の影響はさらに小さくなる．

5.5 簡易ハッシュ値の長さによる処理時間の比較

評価時に採用した簡易ハッシュ値の長さ l_h は 8 bit である．これは，NTMobile の予備フィールドが 8 bit であり，そのフィールドを利用することを想定したためである．本節では，l_h を 8 bit より長くとれるケースにおいて，違いがどのように出るかについて調査した．この調査で

表 5 簡易ハッシュ値の長さと $\overline{P_s}$ の関係
Table 5 Relationship between simple hash value and $\overline{P_s}$.

l_h[bit]	$\overline{P_s}$
8	9.9609×10^{-1}
16	9.9998×10^{-1}
32	9.9999×10^{-1}

表 6 簡易ハッシュサイズと値比較処理時間
Table 6 Size of simple hash value and processing time of value comparison.

	処理時間 [μs]
a) 8 bit	0.575
b) 16 bit	0.532
c) 32 bit	0.520

の測定条件は，OS：Windows8.1 64 bit，CPU：i3-4025U 1.90 GHz，Memory：4.00 GB のホストマシンであり，実験とは独立した環境で行っている．したがって，測定結果の比率のみを考察に利用する．

まず，l_h が簡易認証で破棄される確率 $\overline{P_s}$ に与える影響を調査した．l_h を 8 bit，16 bit，32 bit と変化させたときの $\overline{P_s}$ を，式 (5) により計算した結果を**表 5** に示す．この結果から，l_h が大きくなるほど $\overline{P_s}$ は大きくなっていることが分かる．ただし，不正パケットは MAC 認証で最終的に必ず破棄されるうえ，l_h による $\overline{P_s}$ の変化には大差がないので，ここに多くの領域を要するのは望ましいとはいえない．したがって，この観点からは簡易ハッシュ値は 8 bit で十分であるといえる．

次に，l_h が簡易認証処理時間 t_s に与える影響を調査した．簡易認証処理を，ハッシュ値を求めるまでのハッシュ演算処理と，演算結果とヘッダ内の値とを比較する値比較処理に分離し，それぞれに要する時間の比率を調査した．その結果，ハッシュ演算処理は 70.3%，値比較処理は 29.7%であった．l_h が異なる場合，ハッシュ演算処理ではハッシュ演算の結果の一部を切り取るだけなので，ハッシュ演算処理時間は l_h にかかわらず同じである．一方，値比較処理時間は，簡易ハッシュ値の長さ l_h と，パケット内の対応するフィールドの場所により異なる．

そこで，値比較処理時間に与える影響を調査した．プログラムの制約上，l_h の最小値は 8 bit である．ここでは，簡易ハッシュ値を格納するフィールドとして，次の 3 通りの場合で実測して比較した．すなわち，パケットを処理しやすい 4 Byte 単位で区切り，a) 最後の 1 Byte に割り当てた場合（8 bit），b) 最後の 2 Byte に割り当てた場合（16 bit），c) 4 Byte すべてに割り当てた場合（32 bit）である．なお，a) は実験で用いた NTMobile の場合に相当する．

表 6 に，l_h を変化させた場合の値比較処理時間の測定結果を示す．測定結果から，l_h が 32 bit の場合が最も高速であることが分かった．これは，32 bit（int 型）以外の演算では，演算しやすいようにコンパイラで暗黙の型変換を行っているためと考えられる．

ここで，8 bit と 32 bit の処理時間の差は 10.6%であるが，値比較時間は簡易認証処理の 29.7%を占めるため，簡易認証処理全体での比率に換算すると，この差は 3.1%となる．簡易認証処理が正常時のスループットへ与える影響は 0.95%であるので，8 bit と 32 bit の時間差がスループットに対して与える影響は 0.029%となり，ほとんど無視できる．以上から，簡易認証で破棄される確率に大差がなく，処理時間の差はほとんど無視できるので，簡易ハッシュ値の長さは 8 bit で十分であるといえる．

NTMobile では NTM header 内の 8 bit の予備フィールドを簡易ハッシュフィールドとして利用する方法と，NTM header に 4 Byte を追加し 16 bit または 32 bit の簡易ハッシュ値を格納するエリアを確保する方法が考えられる．ヘッダに 4 バイトを追加する方法は，その分がオーバヘッドとなり，データサイズを 1,000 Byte とすると 0.4%のスループット低下となる．そのため，NTMobile においては簡易ハッシュ値を 8 bit とし，予備フィールドに格納する方法が最適である．

5.6 既存プロトコルへの適用の可能性

本提案方式を既存プロトコルに適用するには，パケット内に 8 bit 以上の簡易ハッシュ値を追加する必要がある．パケットの基本フォーマットを変える必要がある場合は，本提案を適用するのは現実的ではない．パケット内に連続した予備フィールドがあり，そのフィールドが暗号化適用範囲外で，かつパケット認証の範囲内である場合に本提案方式を適用することができる．

NTMobile は上記条件を満たす 8 bit の予備フィールドがあるため適用可能である．現行の IPsec ESP では，該当する予備フィールドがないので適用できない．IPsec AH は該当するフィールドとして 16 bit の予備フィールドがあるため，提案方式を適用可能である．

6. まとめ

本論文では，既存のパケット検証処理に加え，事前に共通鍵とシーケンス番号のみを用いた簡易認証を実行させる方式を提案した．提案方式を用いたパケット検証テストプログラムを作成し，Linux において正常に動作することを確認した．測定とシミュレーションにより，既存方式と比較して不正パケットの検証処理時間を 87%削減できることが分かった．また，正常な処理に与える影響は最大でも 0.95%以下である．簡易認証に用いる簡易ハッシュ値の長さは 8 bit で十分であることを示した．

今回はテストプログラムを用いた実験を行ったが，NTMobile に対する DoS 攻撃耐性の向上が見込めることが分かった．したがって，今後は提案方式を NTMobile に正式

仕様として組み込む予定である．

参考文献

[1] Rescorla, E. and Korver, B.: Guidelines for Writing RFC Text on Security Considerations, RFC 3552, IETF (2003).
[2] Kent, S. and Seo, K.: Security Architecture for the Internet Protocol, RFC 4301, IETF (2005).
[3] Krawczyk, H., Bellare, M. and Canetti, R.: HMAC: Keyed-Hashing for Message Authentication, RFC 2104, IETF (1997).
[4] 鈴木秀和, 上醉尾一真, 水谷智大, 西尾拓也, 内藤克浩, 渡邊 晃：NTMobile における通信接続性の確立手法と実装，情報処理学会論文誌，Vol.54, No.1, pp.367–379 (2013).
[5] 内藤克浩，上醉尾一真，西尾拓也，水谷智大，鈴木秀和，渡邊 晃，森香津夫，小林英雄：NTMobile における移動透過性の実現と実装，情報処理学会論文誌，Vol.54, No.1, pp.380–393 (2013).
[6] 納堂博史，鈴木秀和，内藤克浩，渡邊 晃：NTMobile における自律的経路最適化の提案，情報処理学会論文誌，Vol.54, No.1, pp.394–403 (2013).
[7] 上醉尾一真，鈴木秀和，内藤克浩，渡邊 晃：IPv4/IPv6 混在環境で移動透過性を実現する NTMobile の実装と評価，情報処理学会論文誌，Vol.54, No.10, pp.2288–2299 (2013).
[8] 加古将規，鈴木秀和，内藤克浩，渡邊 晃：NTMobile を無限の規模に拡大できる仮想 IPv4 アドレス管理方式の提案，情報処理学会論文誌，Vol.58, No.3, pp.726–735 (2017).
[9] 金松友哉, 大久保陽平, 山田貴之, 鈴木秀和, 内藤克浩, 渡邊 晃：NTMobile における通信制御機能の提案と実装，電気学会論文誌 C，Vol.137, No.12, pp.1571–1579 (2017).
[10] Kent, S.: IP Authentication Header (AH), RFC 4302, IETF (2005).
[11] Kent, S.: IP Encapsulating Security Payload (ESP), RFC 4303, IETF (2005).
[12] Kaufman, C., Hoffman, P., Nir, Y., Eronnen, P. and Kivinen, T.: Internet Key Exchange Protocol Version 2 (IKEv2), RFC 7296, IETF (2014).
[13] FNV Hash, available from ⟨http://www.isthe.com/chongo/tech/comp/fnv/index.html⟩.
[14] Kent, S. and Atkinson, R.: Security Architecture for the Internet Protocol, RFC 2401, IETF (1998).

鴨下 友馬 （正会員）

2018 年 3 月名城大学理工学部情報工学科卒業．2018 年 4 月株式会社アイ・ティ・ワークス入社．学士（工学）．在学時代は主としてネットワークセキュリティに関する研究に従事．

鈴木 秀和 （正会員）

2004 年 3 月名城大学理工学部情報科学科卒業．2009 年 3 月同大学大学院理工学研究科電気電子・情報・材料工学専攻博士後期課程修了．2008 年 4 月日本学術振興会特別研究員．2010 年 4 月名城大学理工学部助教．2015 年 4 月より同大学理工学部准教授および東北大学電気通信研究所共同研究員を兼務．ネットワークセキュリティ，モバイルネットワーク，ホームネットワーク等の研究に従事．博士（工学）．IEEE，ACM，電子情報通信学会各会員．

内藤 克浩 （正会員）

1977 年生．1999 年 3 月慶應義塾大学理工学部電気工学科卒業．2004 年 3 月名古屋大学大学院工学研究科情報工学専攻博士課程後期課程修了．2004 年 4 月三重大学工学部電気電子工学科助手．2007 年 4 月同大学助教．2011 年 9 月カリフォルニア大学ロサンゼルス校客員研究員．2014 年 4 月愛知工業大学情報科学部准教授．2016 年情報処理学会・長尾真記念特別賞受賞．博士（工学）．IEEE，電子情報通信学会各会員．主として無線ネットワーク，モバイルコンピューティングの研究に従事．

渡邊 晃 （正会員）

1974 年慶應義塾大学工学部電気工学科卒業．1976 年同大学大学院工学研究科修士課程修了．同年三菱電機株式会社入社後，LAN システムの開発・設計に従事．1991 年同社情報技術総合研究所に移籍し，ルータ，ネットワークセキュリティ等の研究に従事．2002 年名城大学理工学部教授，現在に至る．博士（工学）．電子情報通信学会，IEEE 各会員．