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

140441043 鴨下 友馬
渡邊研究室

1. はじめに

モバイルネットワークの普及に伴い，ネットワークセキュリティを脅かす DoS 攻撃 (Denial of Service Attack) が問題となっている．DoS 攻撃対策の一例として，共通鍵を事前に共有している場合は HMAC (Hash-based Message Authentication Code) を用いたパケット認証（以下，MAC 認証）を利用することができる．しかし，この認証方式ではパケット長が長いと処理時間が長くなる．

そこで，共通鍵とシーケンス番号のみを用いた簡易認証方式を提案する．この方式では，共通鍵とシーケンス番号から生成した短いハッシュ値をパケット内に付加し，その値を最初に検証する．これにより，不正パケットのほとんどを高速に検出することが可能となる．

本稿では，実験による評価を行い，提案方式の有用性を示す．実験においては，移動透過性と通信接続性の両者を同時に実現する NTMobile (Network Traversal with Mobility) [1] を用いた．

2. 既存のパケット検証処理

IPsec (IP security) の ESP (Encapsulating Security Payload) における，パケット受信時の DoS 攻撃防止処理を説明する．パケット検証はリプレイ攻撃チェック，MAC 認証の順に行い，不正パケットであると判定した場合にはその時点で破棄を決定し，以降の処理は行わない．ここで，リプレイ攻撃とは攻撃者が正規のパケットを盗聴し，それを再送する攻撃である [2]．リプレイ攻撃チェックはこれを阻止するための処理であり，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ことで，リプレイ攻撃パケットを検出する．リプレイ攻撃では過去に送信された正規のパケットを攻撃に利用するため，MAC 認証では検出できず，リプレイ攻撃チェックは必須である．MAC 認証まで成功した場合は正規のパケットとみなし，リプレイ防御ウィンドウの更新を行う．

NTMobile は，移動透過性と通信接続性の両者を同時に実現する技術であり，IPsec と同様にエンドツーエンドのセキュリティを実現することができる．NTMobile のパケット検証は ESP と同様で，リプレイ攻撃チェック，MAC 認証の順に行い，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．

3. 提案方式

提案方式では，送信側は，通信に用いる共通鍵とシーケンス番号を用いて 8bit のハッシュ値（以下，簡易ハッシュ値）を生成してパケットに付与する．ハッシュ関数は，演算時間の短い FNV-1(32bit) を用いることとする．受信側は，最初に簡易ハッシュ値を計算し，受信パケットに格納された値と比較する．これらの値が不一致であれば不正パケットであると判定して破棄し，一致していれば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で，リプレイ攻撃チェック，MAC 認証の順に行い，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．提案方式は ESP，NTMobile 双方に適用できるが，パケット内に，簡易認証に係るフィールド（8bit）を追加する必要がある．

4. 実験と評価

4.1 検証時間の測定

MAC 認証範囲を 1,036Byte としてパケット検証処理を 100,000 回実行した際の，処理時間の平均値を表 1 に示す．

表 1: 検証時間の実測値

$t_s[\mu s]$	$t_r[\mu s]$	$t_m[\mu s]$
0.536	0.414	3.835

表 2: 不正パケット検出率

	簡易認証あり	簡易認証なし
$\overline{P_s}$	9.961×10^{-1}	0
$\overline{P_r}$	1.819×10^{-12}	4.657×10^{-10}
$\overline{P_m}$	3.906×10^{-3}	9.999×10^{-1}

表 3: 正規のパケットの検証時間

$t_u[\mu s]$	簡易認証なし $[\mu s]$	簡易認証あり $[\mu s]$
0.561	4.810	5.346

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 認証に要する時間である．

4.2 不正パケットの検証時間

不正パケットが簡易認証で破棄される確率，リプレイ攻撃チェックで破棄される確率，MAC 認証で破棄される確率をそれぞれ $\overline{P_s}$, $\overline{P_r}$, $\overline{P_m}$ とすると，不正パケットの検証時間の平均値 E は式 (1) のようになる．

$$E = t_s\overline{P_s} + (t_s + t_r)\,\overline{P_r} + (t_s + t_r + t_m)\,\overline{P_m} \tag{1}$$

これに表 1 の実測値，および表 2 の不正パケット検出率を適用すると，

$$E = 0.553\,[\mu s] \tag{2}$$

となった．一方，簡易認証を適用しない場合は，

$$E|_{\overline{P_s}=0,\,t_s=0} = 4.249\,[\mu s] \tag{3}$$

となる．この結果から，簡易認証により，不正パケットの検証時間を約 1/8 に短縮でき，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．

4.3 正規のパケットの検証時間

攻撃がない状態においては，パケット検証に常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，この増加した負荷が全体の処理時間に与える影響を調査した．正規のパケットの検証処理を 100,000 回実行した際の，リプレイ防御ウィンドウ更新処理時間の平均値 t_u と，全体の処理時間の理論値を表 3 に示す．提案手法では t_s が加わるため全体の検証処理時間は若干長くなるものの，この後の復号処理時間（MAC 認証の約 10 倍），さらには正規の受信処理時間を考慮すると影響は極めて小さいと言える．

5. まとめ

本稿では，共通鍵とシーケンス番号のみを用いた簡易認証方式を提案し，実験によりその有用性を示した．今後，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] 上醉尾一真ほか: 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).

[2] Rescorla, E., et al: Guidelines for Writing RFC Text on Security Considerations, RFC 3552, IETF (2003).

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

渡邊研究室
140441043
鴨下 友馬

Watanabe Lab.

Watanabe Lab.

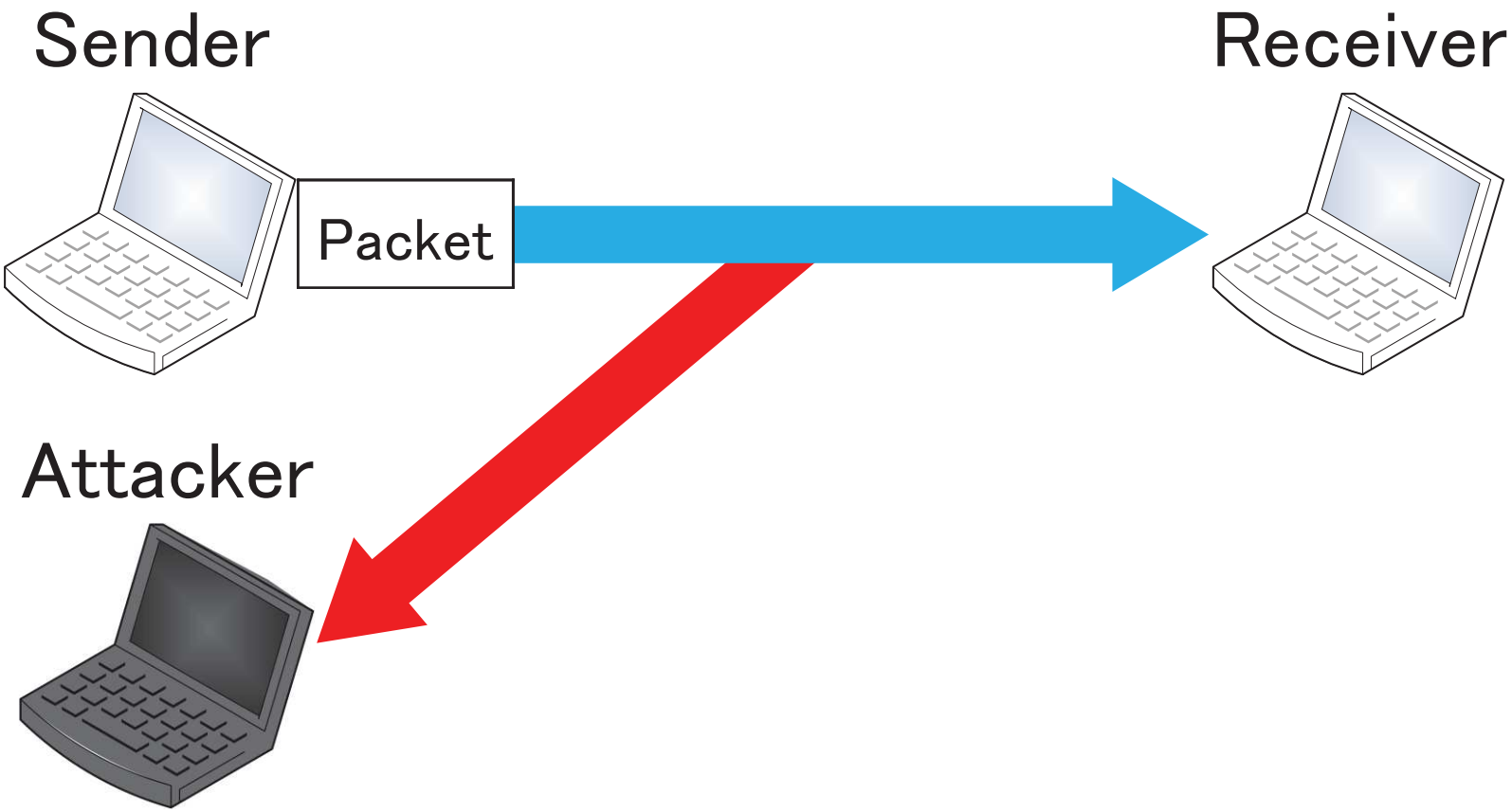
研究背景

- 移動通信端末の普及
- インターネット利用の需要増加

- ネットワークセキュリティへの脅威
  - 暗号化通信においても有効な攻撃
    - リプレイ攻撃 (Replay Attack)
    - DoS攻撃 (Denial of Service Attack)

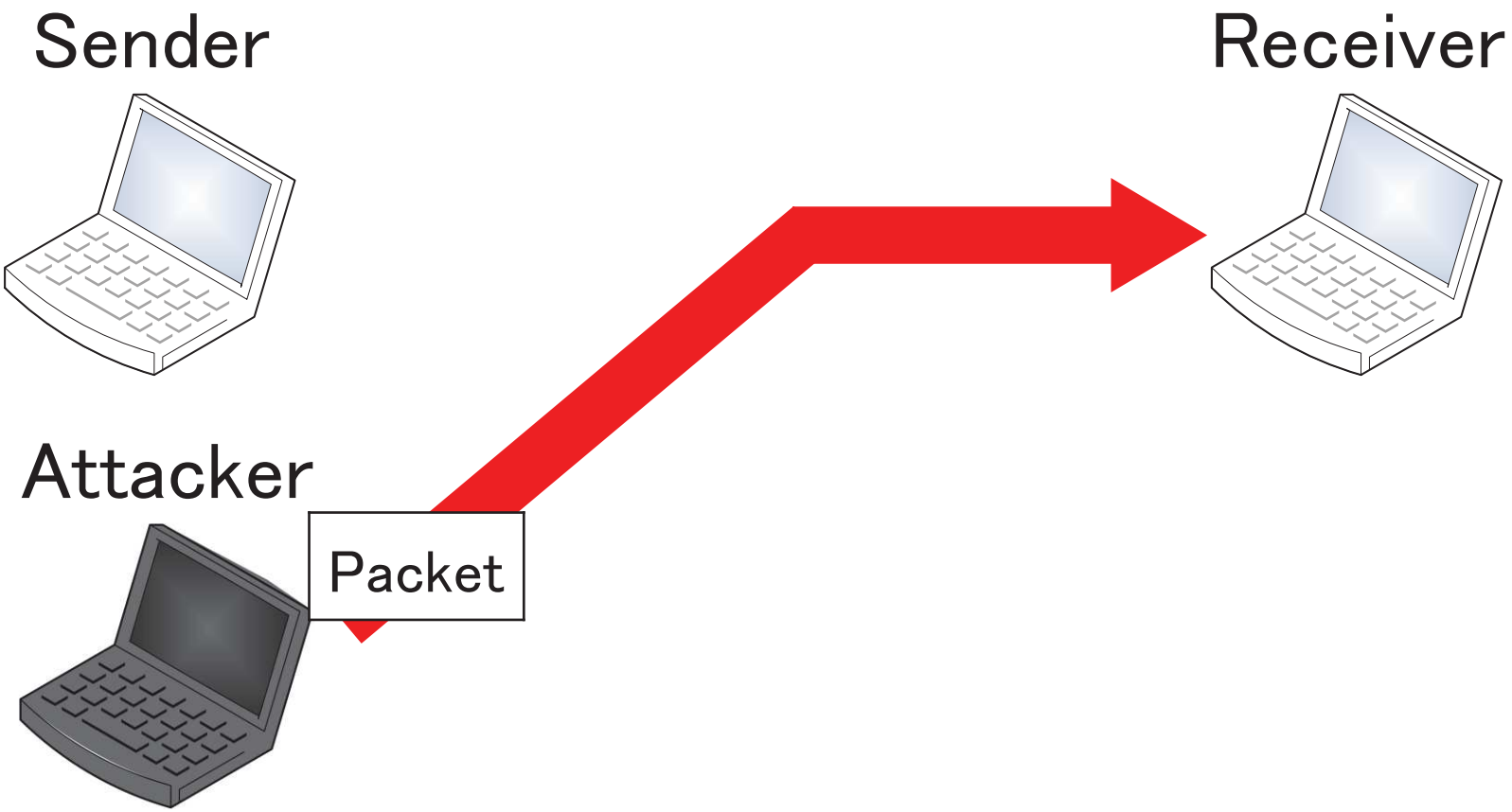
リプレイ攻撃

■過去に送信された正規のパケットを再送する攻撃



リプレイ攻撃

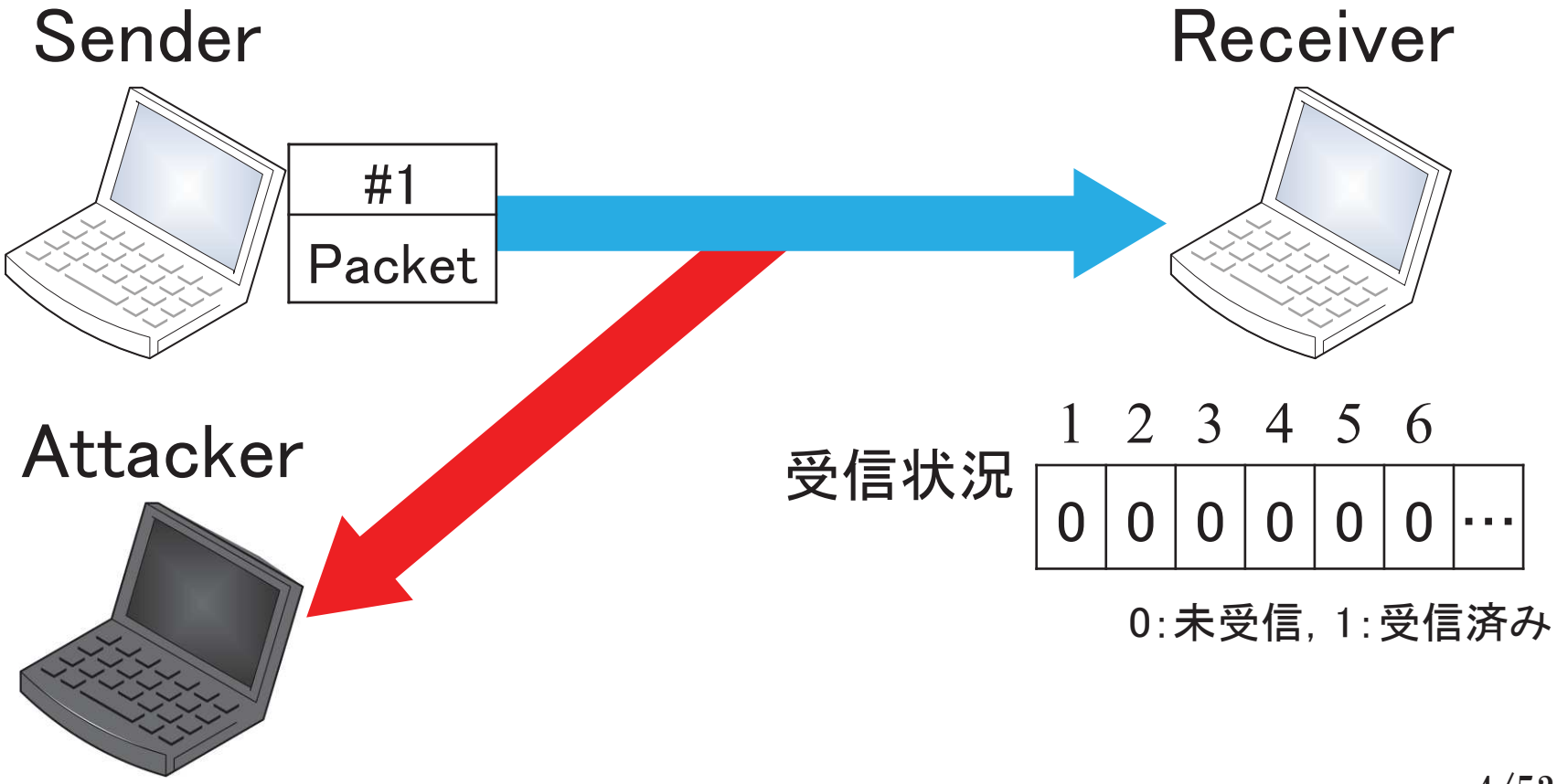
- 過去に送信された正規のパケットを再送する攻撃



Watanabe Lab.

リプレイ攻撃対策:リプレイ攻撃チェック

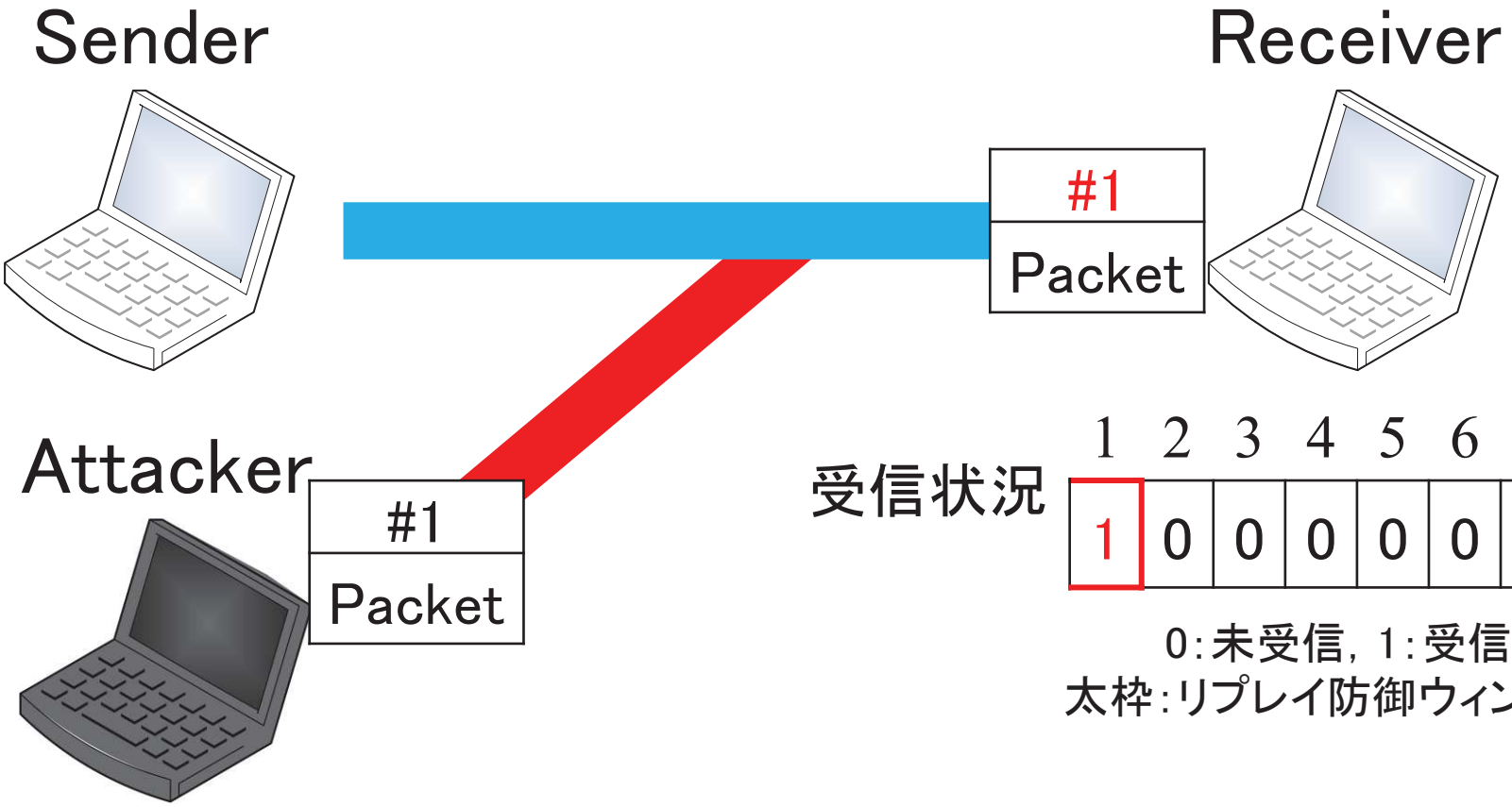
- RFC2401 (IPsec Ver.2)で標準化
- シーケンス番号の使用



Watanabe Lab.

リプレイ攻撃対策:リプレイ攻撃チェック

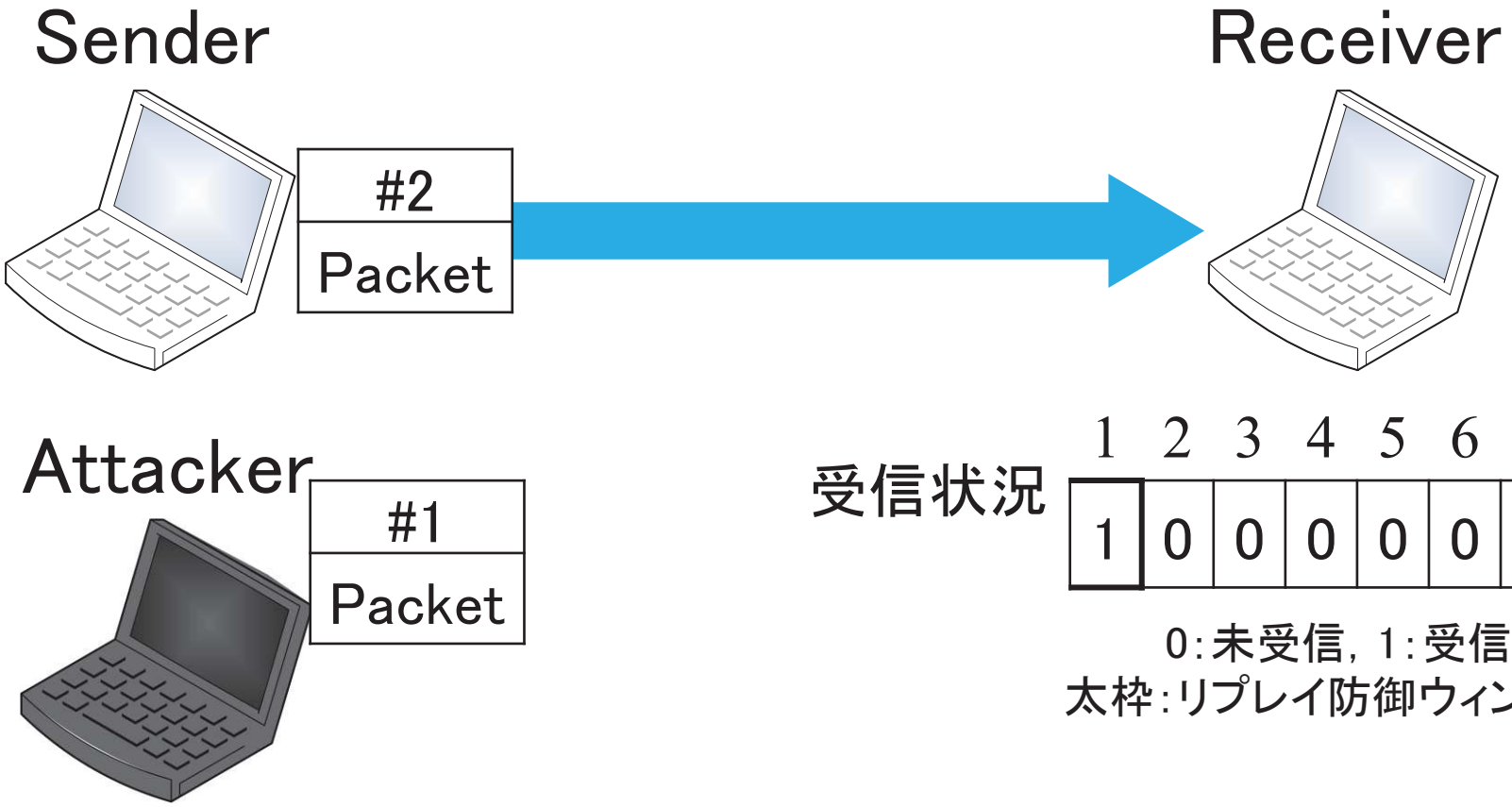
- リプレイ防御ウィンドウにより受信範囲を決定
  - ウィンドウ内か最新で，未受信のシーケンス番号を受理



Watanabe Lab.

リプレイ攻撃対策:リプレイ攻撃チェック

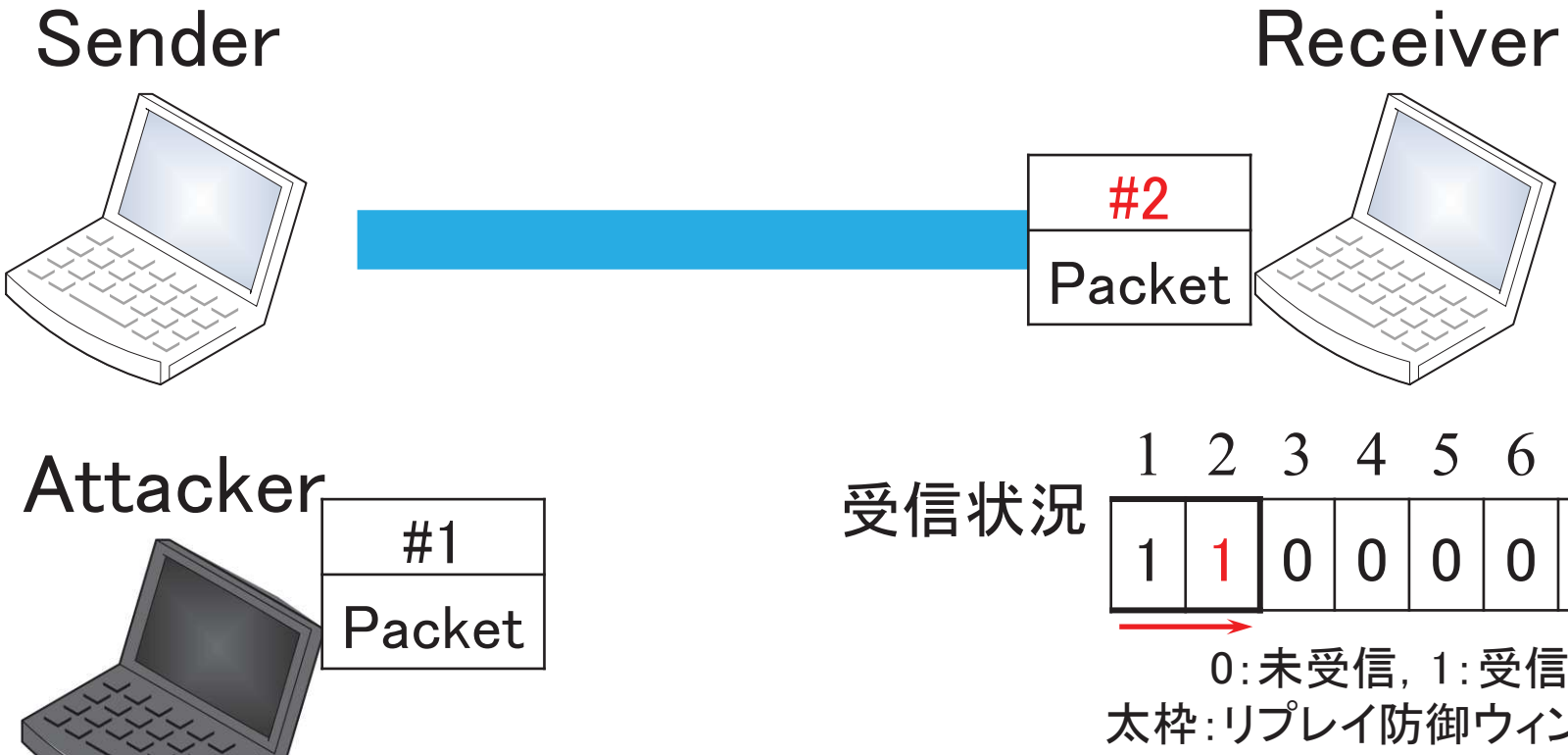
- リプレイ防御ウィンドウにより受信範囲を決定
  - ウィンドウ内か最新で，未受信のシーケンス番号を受理



Watanabe Lab.

リプレイ攻撃対策:リプレイ攻撃チェック

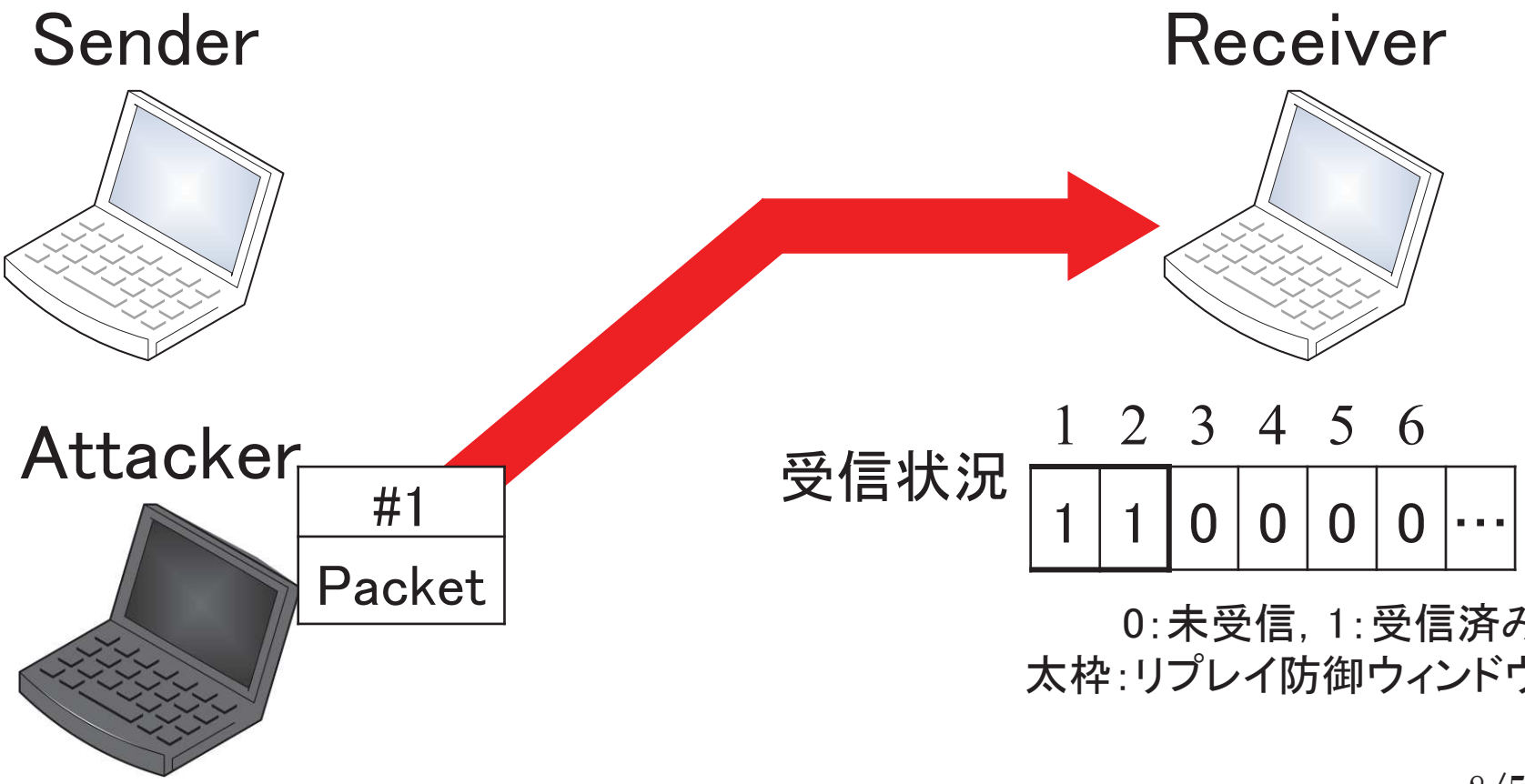
- リプレイ防御ウィンドウにより受信範囲を決定
  - ウィンドウ内か最新で，未受信のシーケンス番号を受理



Watanabe Lab.

リプレイ攻撃対策:リプレイ攻撃チェック

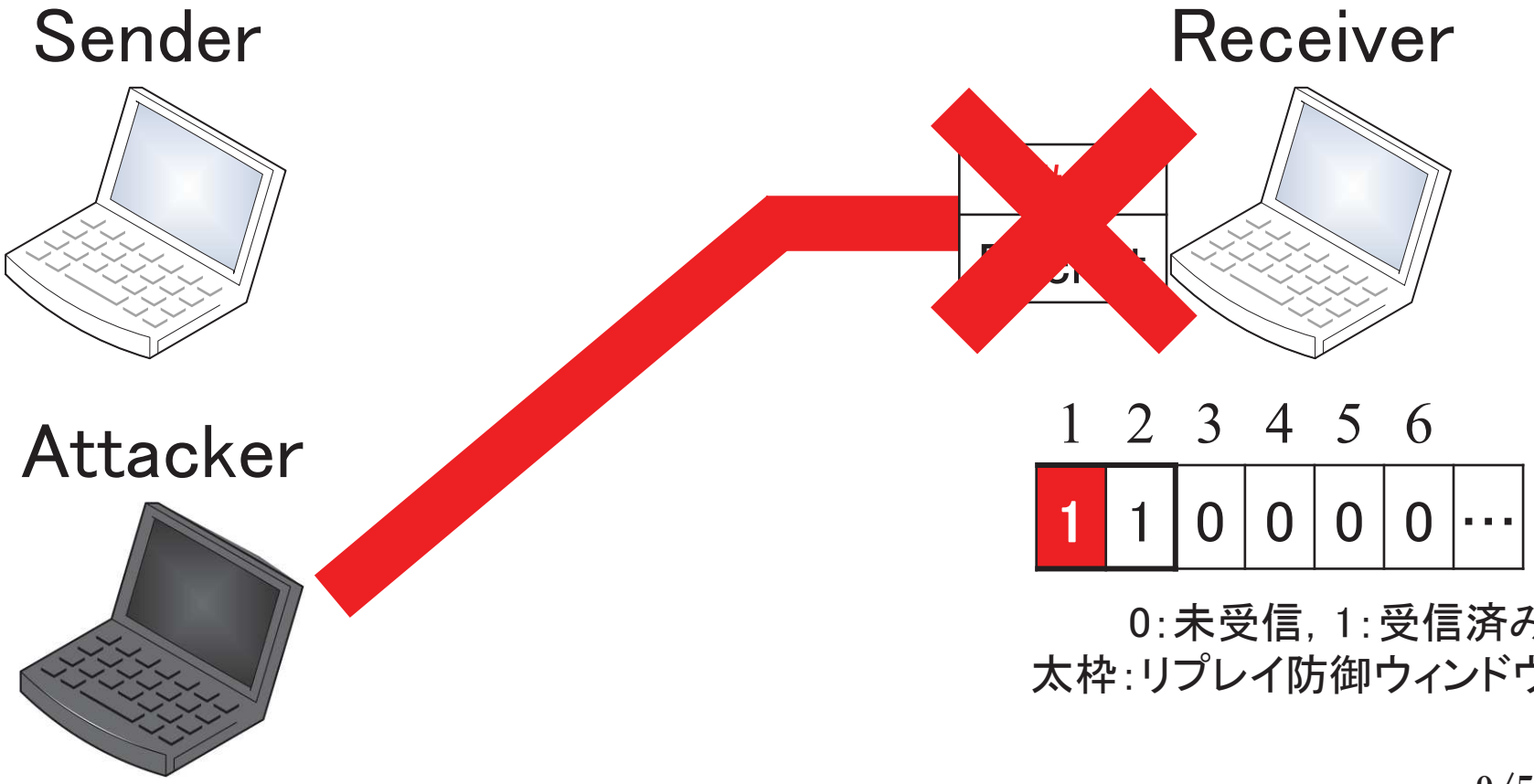
■リプレイ攻撃実施



Watanabe Lab.

リプレイ攻撃対策:リプレイ攻撃チェック

- リプレイ防御ウィンドウにより受信済みと判定
  - → リプレイ攻撃とみなして破棄



DoS攻撃

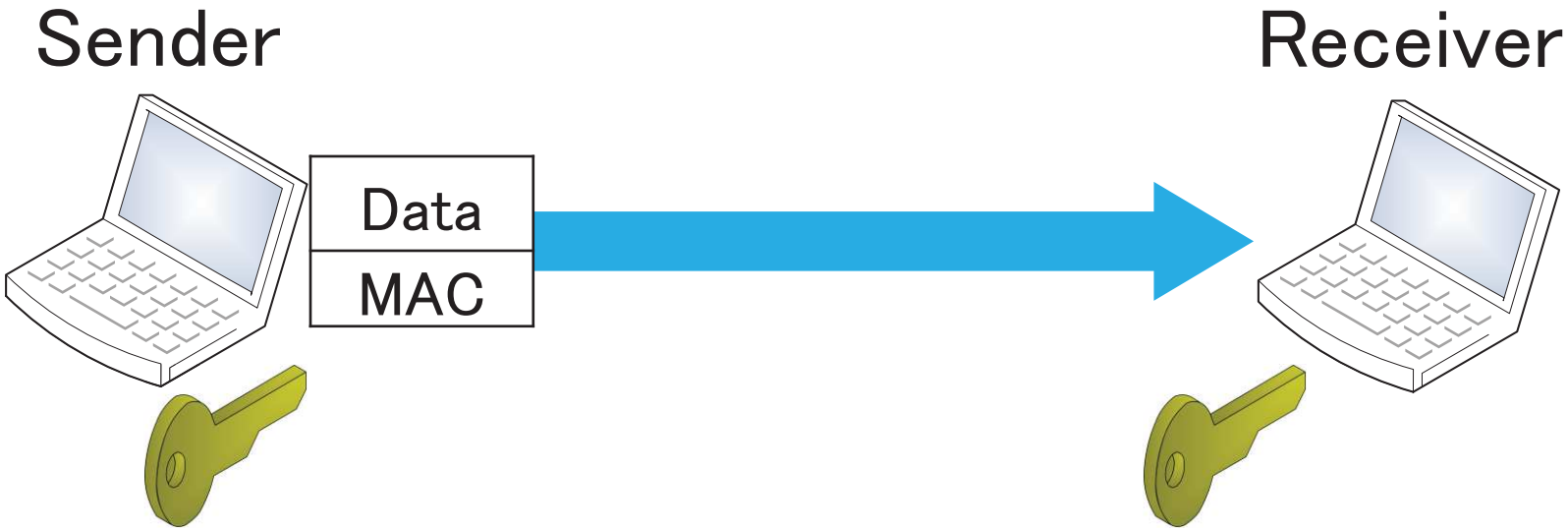
- 受信側に過剰な負荷を与えることで
  サービスを不能にさせる攻撃

DoS攻撃対策

- MAC認証
  - MAC (Message Authentication Code)を用いたパケット検証
  - パケットのデータ部と共通鍵を用いてMACを生成

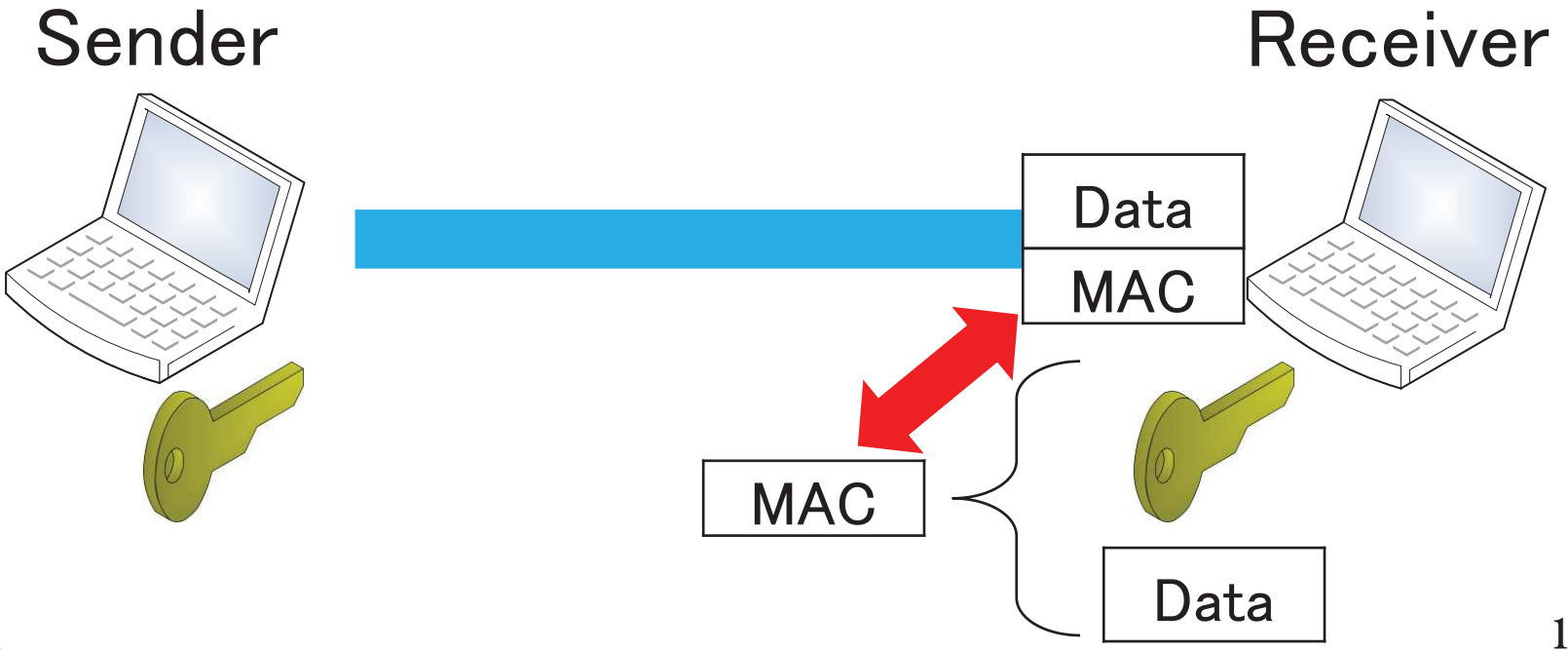
DoS攻撃対策

- MAC認証
  - MAC (Message Authentication Code)を用いたパケット検証
  - パケットのデータ部と共通鍵を用いてMACを生成



DoS攻撃対策

- MAC認証
  - MAC (Message Authentication Code)を用いたパケット検証
  - 受信側はMACを生成してパケットのMACと比較



Watanabe Lab.

既存技術：IPsec

- ESP (Encapsulating Security Payload)
  - OSI参照モデルのネットワーク層（第3層）における一般的なセキュリティプロトコル（RFC4303）
  - リプレイ攻撃チェック，MAC認証の順に行う
    - リプレイ攻撃パケットはMAC認証では検出できない
    - リプレイ攻撃チェックの方が処理時間が速い

Watanabe Lab.

既存技術：NTMobile（オリジナル）

- NTMobile (Network Traversal with Mobility) *1 *2
  - 移動透過性と通信接続性の両者を同時に実現する技術
    - 移動透過性：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性質
    - 通信接続性：ネットワーク環境に関わらず通信を開始することができる性質
  - IPsecと同様のセキュリティを備える
    - 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

*1 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).

*2 納堂博史，八里栄輔，鈴木秀和，内藤克浩，渡邊　晃：実用化に向けたNTMobile フレームワークの実装と評価，第82 回MBL・第53 回UBI 合同研究発表会，No. 46, pp. 1–8 (2017).

Watanabe Lab.

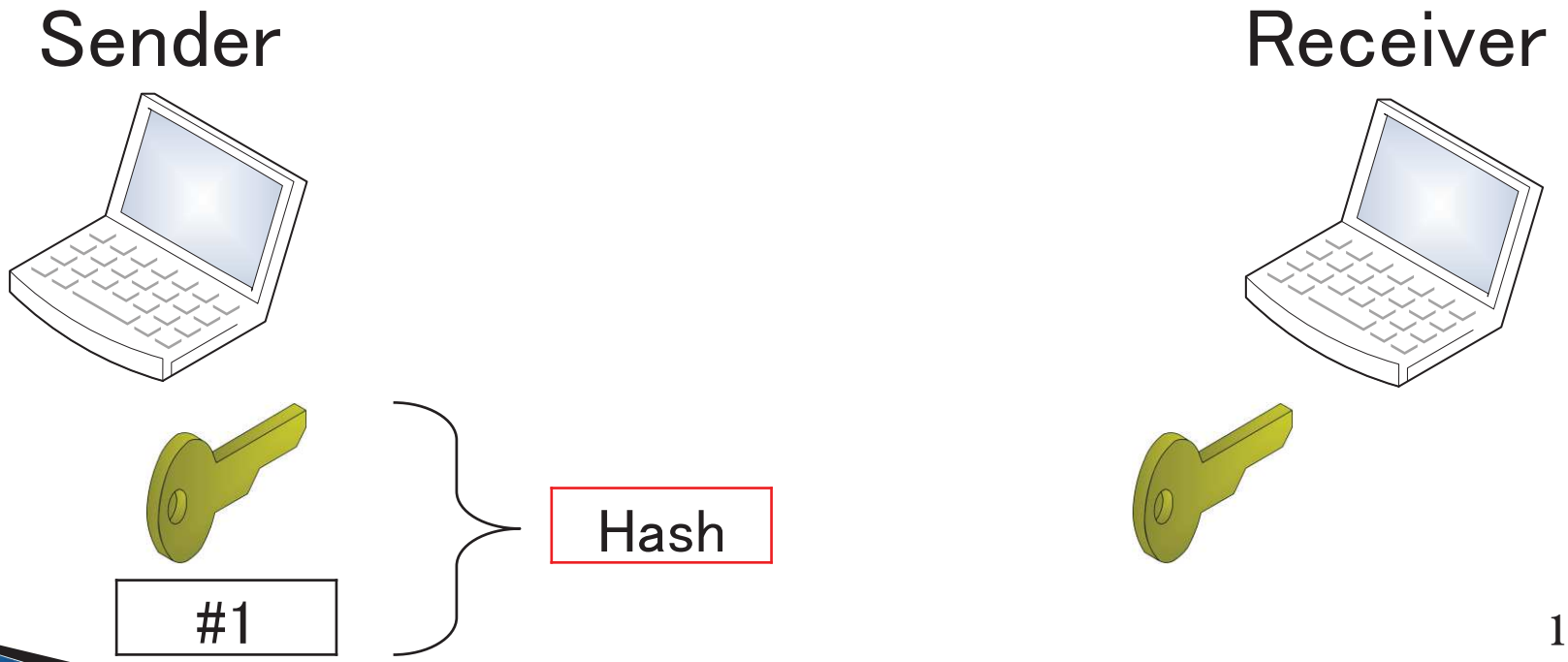
既存技術の課題・要求

- 既存のパケット検証の課題
  - パケット長が長いほどMAC認証の処理時間が長くなる
    - パケット検証処理の大半がMAC認証

→ 大量のパケットを処理するサーバ等では
DoS攻撃を防御するために
少しでも速く不正パケットを検出したい

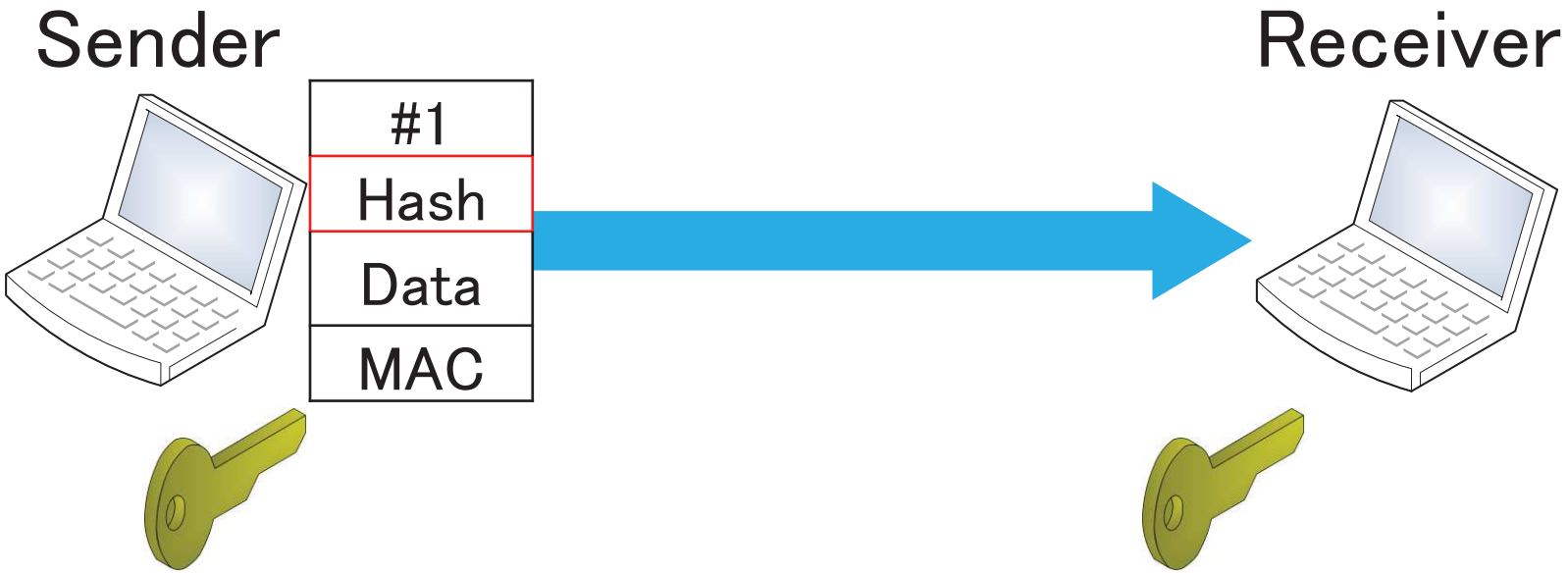
簡易認証方式

- 簡易認証
  - シーケンス番号と共通鍵から簡易ハッシュ値(8bit)を生成



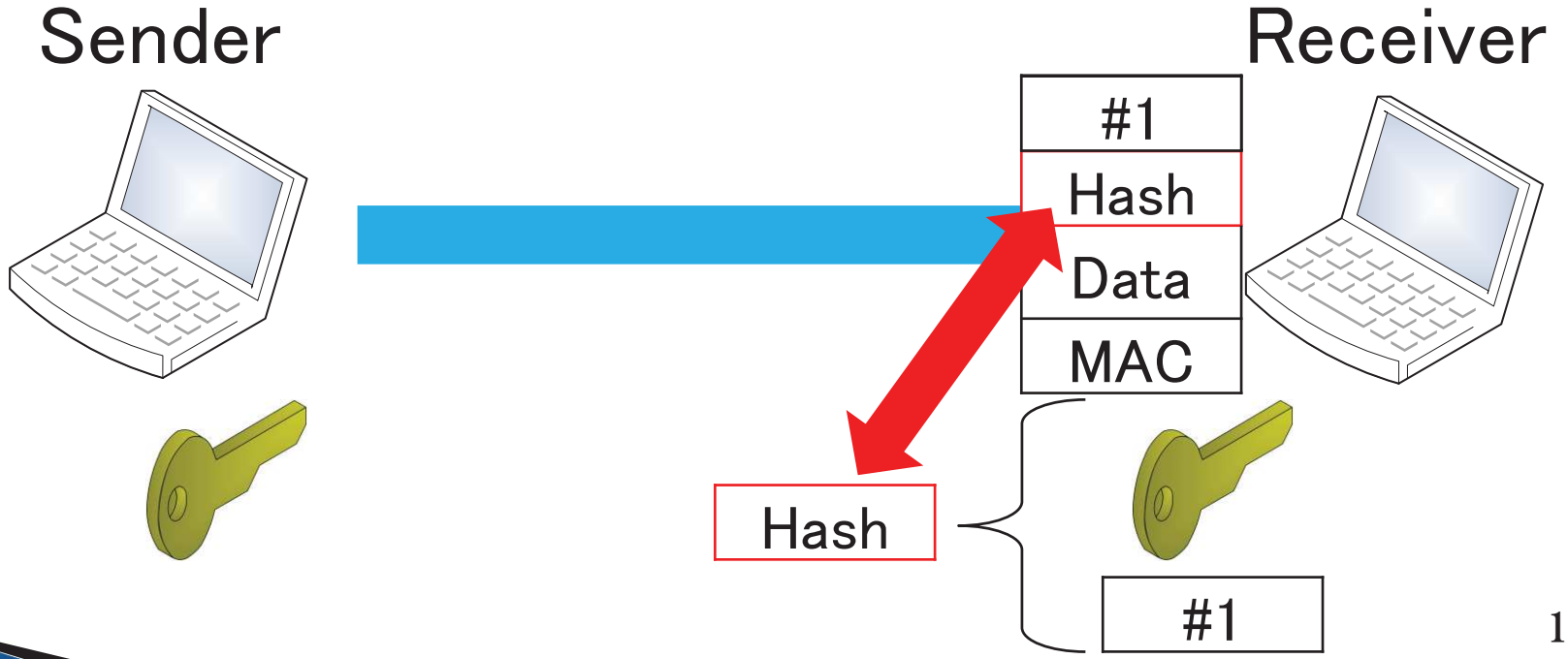
簡易認証方式

- 簡易認証
  - 生成した簡易ハッシュ値をパケットに付加して送信
    - 簡易ハッシュ値を格納するフィールド(8bit)が必要

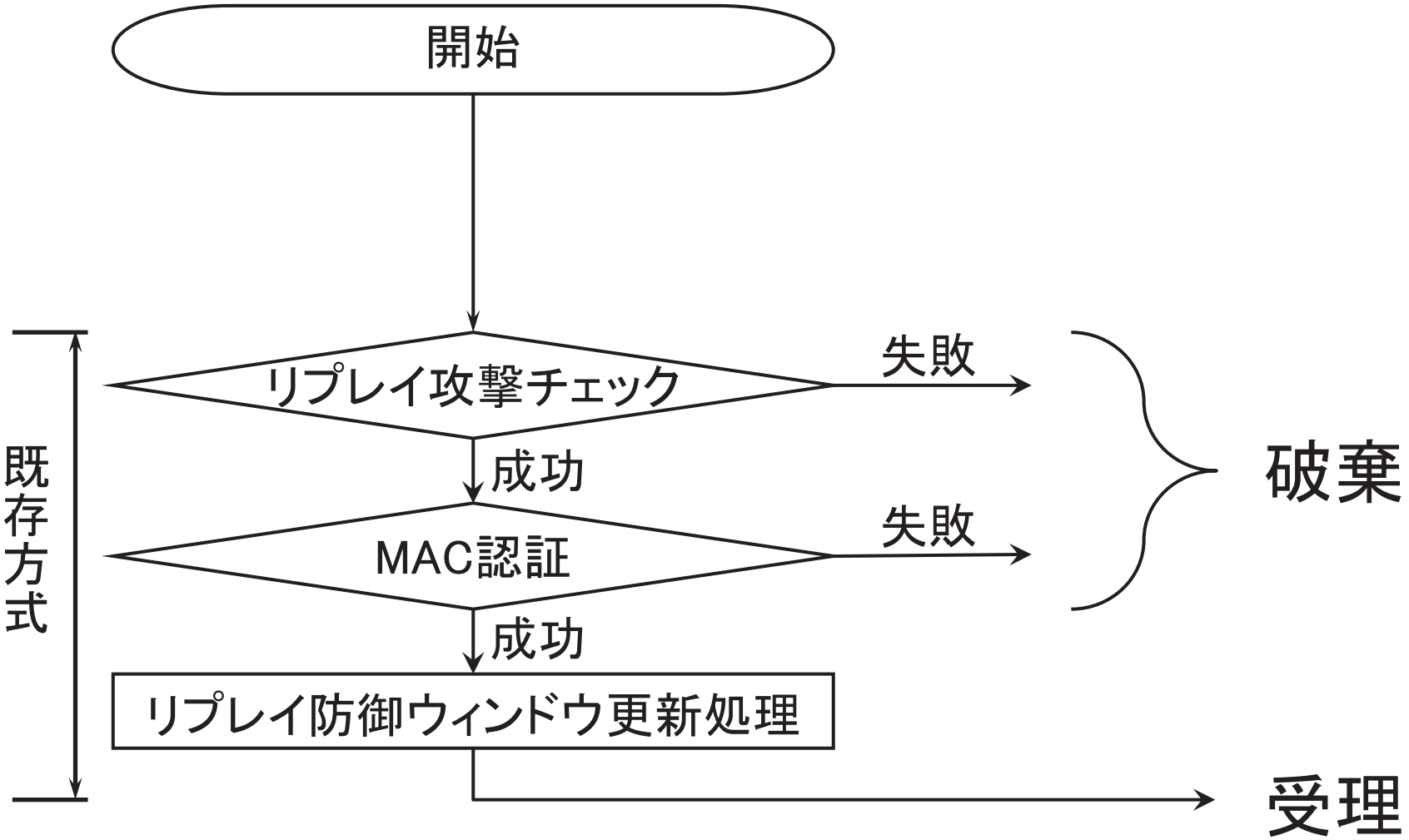


簡易認証方式

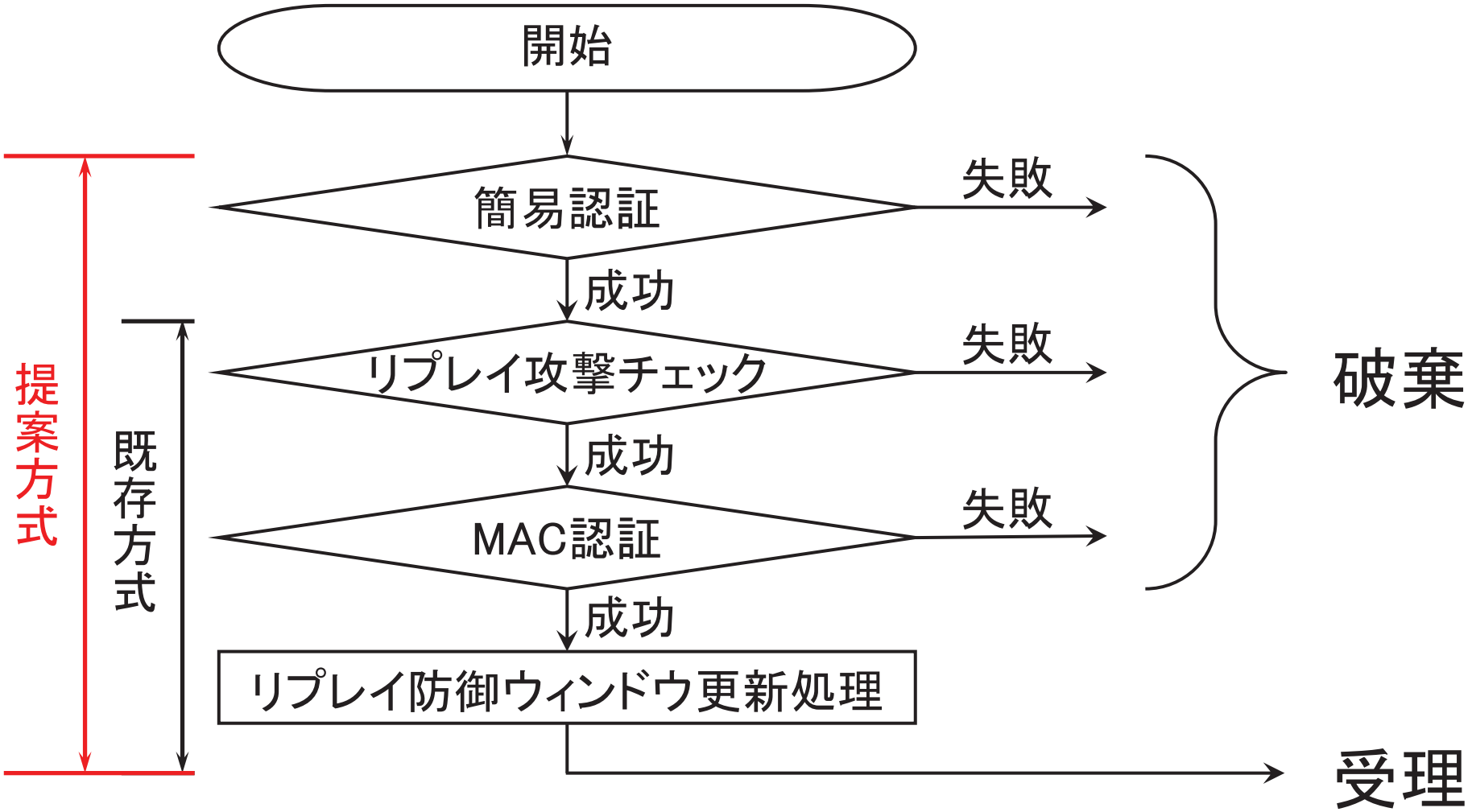
- 簡易認証
  - パケット検証処理の最初に実施
  - 簡易ハッシュ値を生成して
    受信パケットの簡易ハッシュ値と比較



パケット検証順序



パケット検証順序


開始
簡易認証
失敗
成功
リプレイ攻撃チェック
失敗
成功
MAC認証
失敗
成功
リプレイ防御ウィンドウ更新処理
破棄
受理
提案方式
既存方式

Watanabe Lab.

簡易ハッシュ値の生成

- ハッシュ関数はFNV-1 32bit版を使用
  - 入力Mは8bit整数の配列
  - 出力hashは32bit整数
    - このうち，下位8bitを簡易ハッシュ値とする
  - 定数Offset, Primeは32bit整数

```
Offset = 2166136261;
Prime = 16777619;

hash = Offset;
for i := 0 to i < |M| do begin
      hash = (Prime * hash) ^ M[i]
end;
```

Watanabe Lab.

簡易ハッシュ値の生成

- シーケンス番号を8bit毎に分割
  - シーケンス番号が32bitの場合

$$N = \{n_1, n_2, n_3, n_4\}$$

- 共通鍵を8bit毎に分割
  - 共通鍵が128bitの場合

$$K = \{k_1, k_2, \cdots, k_{16}\}$$

- ハッシュ関数への入力Mを生成

$$M = \{N, K\}$$

Watanabe Lab.

動作検証

- テストプログラムの仕様
  - 使用言語：C
  - 通信は行わず，パケット検証処理のみを実行
- 装置の仕様

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

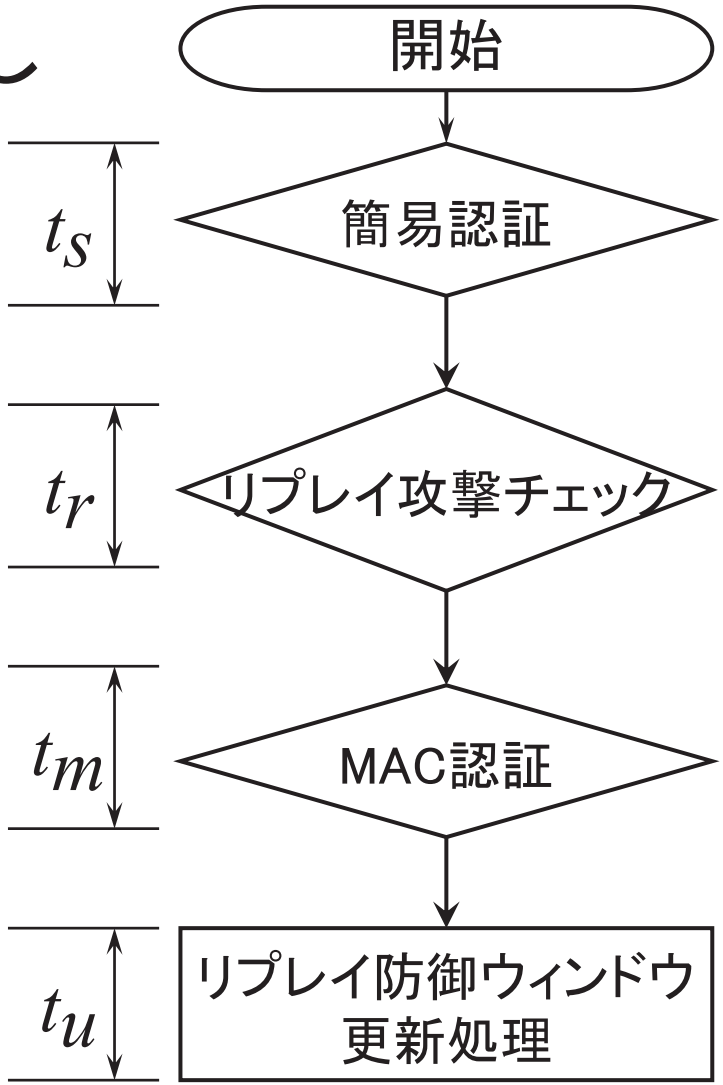
- 以上の条件にて正常に動作することを確認

パケット検証処理時間

- 処理時間t_S, t_r, t_m, t_uを定義し測定
  - 100,000回実行時の平均値

t_S[μs]	t_r[μs]	t_m[μs]	t_u[μs]
0.536	0.414	3.835	0.561

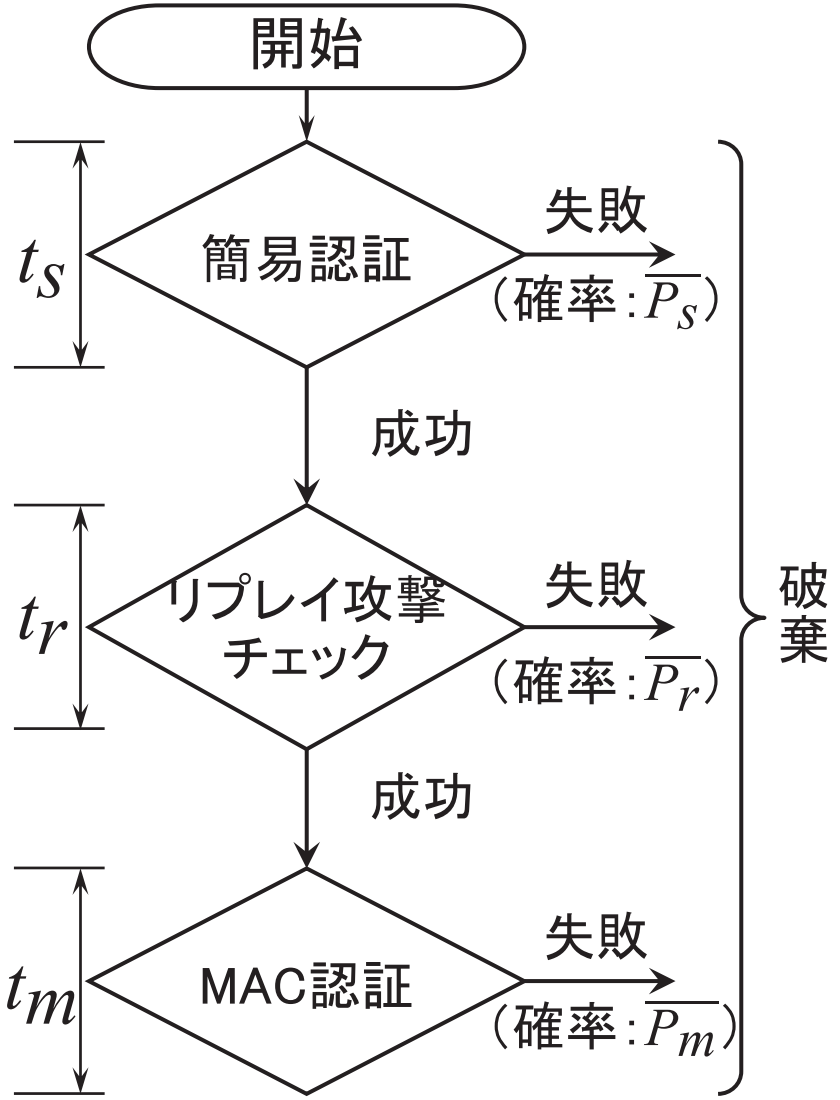
  - MAC生成にはHMAC-MD5を使用
  - MAC生成（パケット認証）の範囲は1036Byte



不正パケットの検証処理時間

- いずれかの処理で破棄されるものを不正パケットとみなす
- 確率$\overline{P_S}, \overline{P_r}, \overline{P_m}$を定義すると,

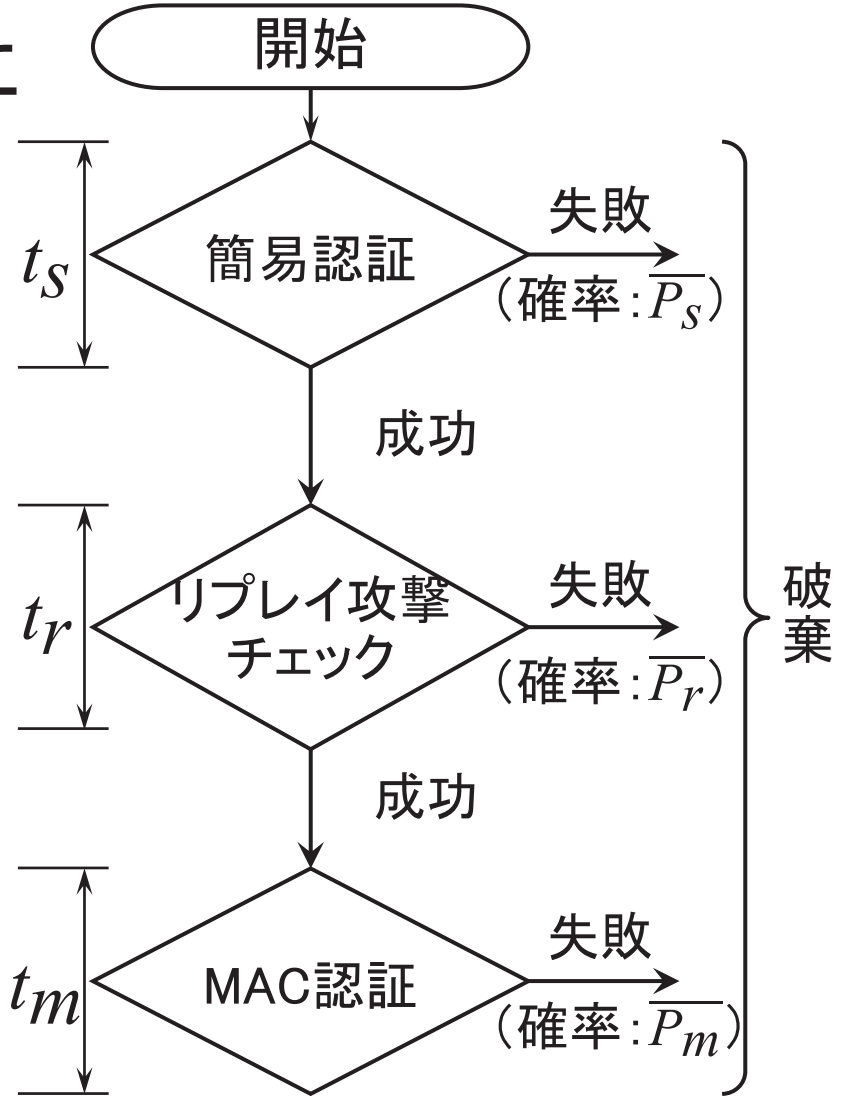
$$\overline{P_S} + \overline{P_r} + \overline{P_m} = 1$$



不正パケットの検証処理時間

- 不正パケットの検証処理に要する時間の平均値は

$$E = t_S \overline{P_S} + (t_S + t_r)\overline{P_r} + (t_S + t_r + t_m)\overline{P_m}$$



Watanabe Lab.

不正パケットの検証処理時間(提案方式)

$$E = t_S \overline{P_S} + (t_S + t_r)\overline{P_r} + (t_S + t_r + t_m)\overline{P_m}$$

- 以下の理論値，実測値を用いる
  - $\overline{P_S}, \overline{P_r}, \overline{P_m}$はMAC認証に到達する確率が最大となるときの理論値，t_S, t_r, t_mは実測値

$$\overline{P_S} = 9.961\times10^{-1}$$

$$\overline{P_r} = 1.819\times10^{-12}$$

$$\overline{P_m} = 3.906\times10^{-3}$$

t_S[μs]	t_r[μs]	t_m[μs]
0.536	0.414	3.835

- このとき，

$$E = 0.553[\mu s]$$

Watanabe Lab.

不正パケットの検証処理時間(既存方式)

$$E = t_S \overline{P_S} + (t_S + t_r)\overline{P_r} + (t_S + t_r + t_m)\overline{P_m}$$

- 既存方式では，簡易認証を使用していないため以下のようになる
  - $\overline{P_r}$, $\overline{P_m}$はMAC認証に到達する確率が最大となるときの理論値，t_r, t_mは実測値

$$\overline{P_S} = 0$$
$$\overline{P_r} = 4.657\times10^{-10}$$
$$\overline{P_m} = 9.999\times10^{-1}$$

t_S[μs]	t_r[μs]	t_m[μs]
0	0.414	3.835

- したがって，

$$E\Big|_{\overline{P_S}=0,\ t_S=0} = 4.249[\mu s]$$

Watanabe Lab.

不正パケットの検証処理時間

提案方式: $E = 0.553[\mu s]$

既存方式: $E\big|_{\overline{P_S} = 0,\ t_S = 0} = 4.249[\mu s]$

- 以上の結果より，提案方式により
  不正パケットの検証処理時間を最大1/8程度に
  短縮することができる
  → 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

Watanabe Lab.

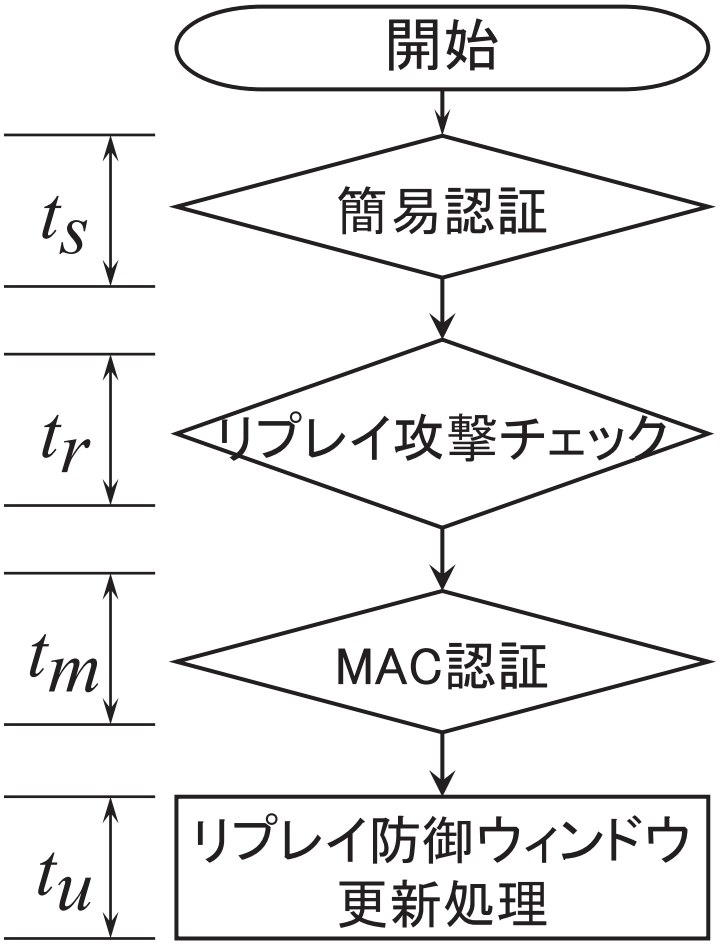
正規のパケットの検証処理時間

- 正規のパケットの場合は，簡易認証の追加分だけ負荷が増加する

t_s[μs]	t_r[μs]	t_m[μs]	t_u[μs]
0.536	0.414	3.835	0.561

既存方式：4.810[μs]

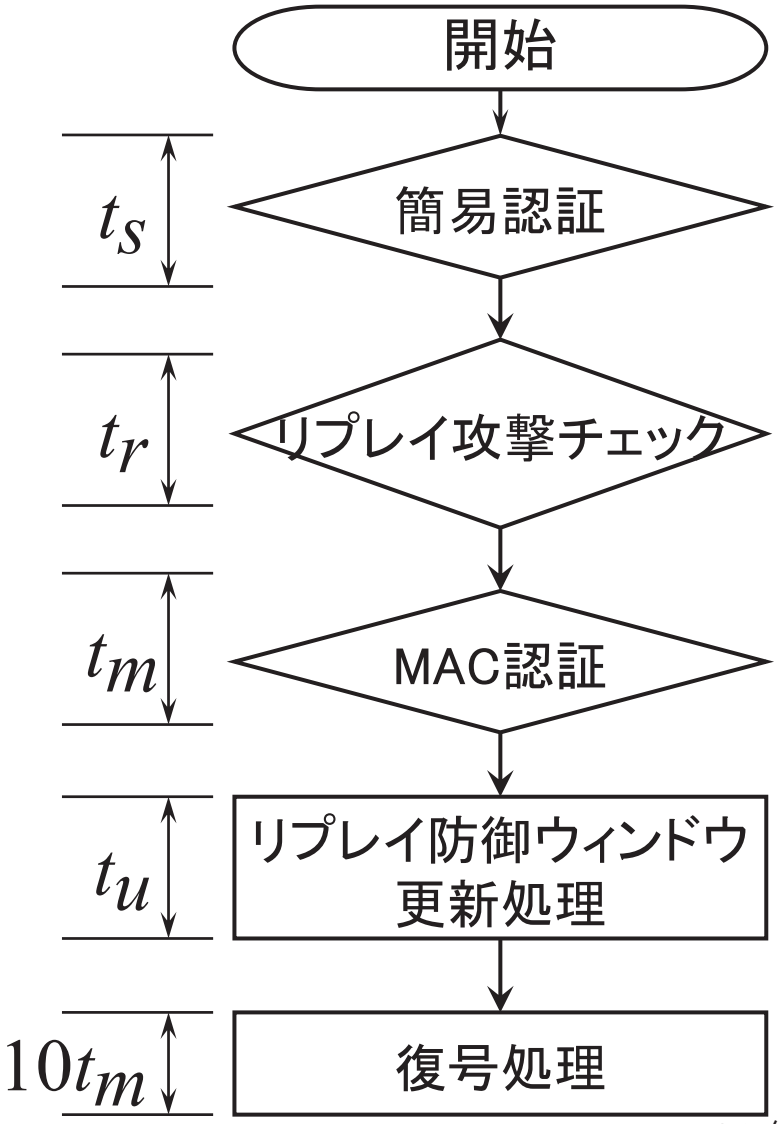
提案方式：5.346[μs]



正規のパケットの検証処理時間

- この後の復号処理は，MAC認証の約10倍の時間を要する

簡易認証の負荷は極めて小さい



Watanabe Lab.

簡易ハッシュ値の長さによる比較

- 簡易ハッシュ値の長さl_hを変化させた場合
  - プログラムの制約上，最小は8bit

l_h[bit]	$\overline{P_s}$	t_s[μs]	E[μs]
8	9.9609×10^{-1}	0.536	0.553
16	9.9998×10^{-1}	0.675	0.675
32	9.9999×10^{-1}	0.823	0.823

  - l_hが大きくなるほど$\overline{P_s}\approx 1$，すなわち$\overline{P_r}\approx 0$, $\overline{P_m}\approx 0$となるので以下が成立

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

Watanabe Lab.

簡易ハッシュ値の長さによる比較

- 簡易ハッシュ値の長さl_hを変化させた場合
  - プログラムの制約上，最小は8bit

l_h[bit]	$\overline{P_S}$	t_S[μs]	E[μs]
8	9.9609×10^{-1}	0.536	0.553
16	9.9998×10^{-1}	0.675	0.675
32	9.9999×10^{-1}	0.823	0.823

  - t_S, Eの増加量に対して$\overline{P_S}$は大差ない

    → 簡易認証の効果が大きく上昇するわけではない

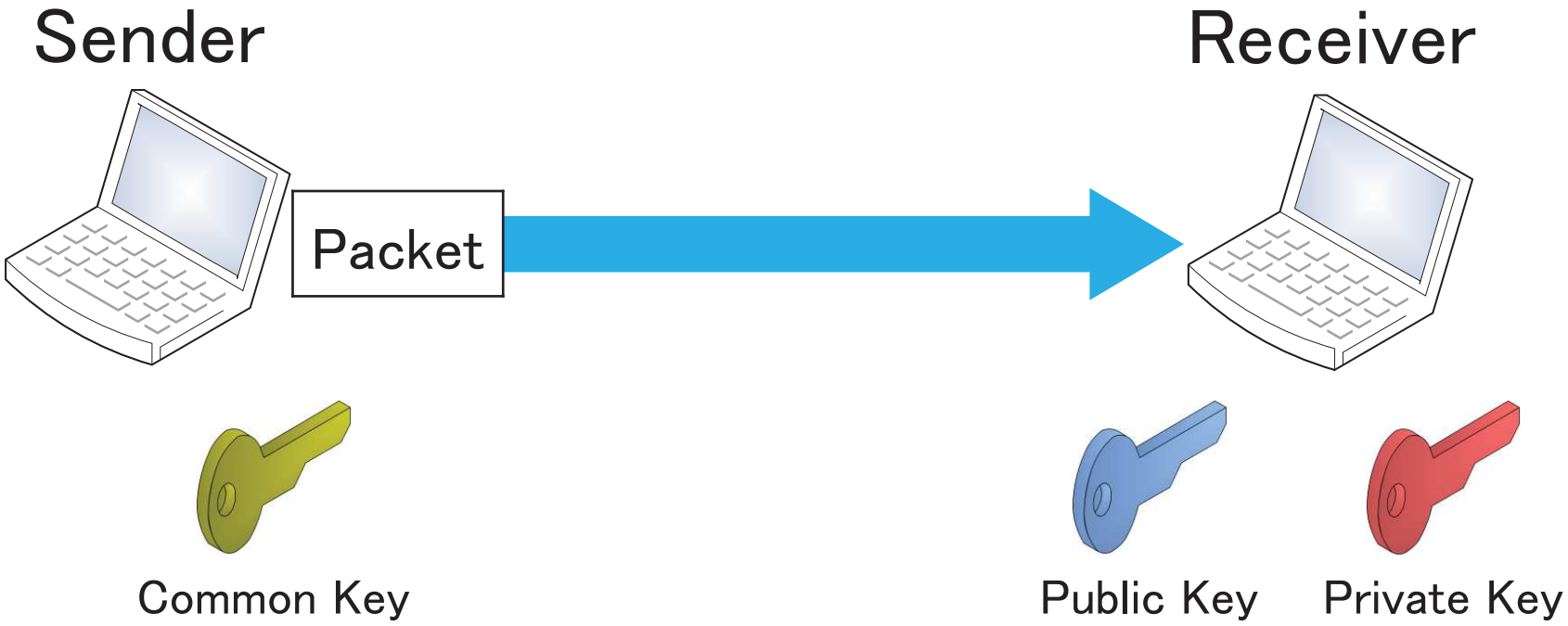
    → 簡易ハッシュ値は8bitが最適

まとめ

- 簡易認証方式の提案
  - 共通鍵とシーケンス番号を使用
  - 既存方式と比較して不正パケットの検証処理時間を最大1/8程度に短縮することが可能
  - 簡易ハッシュ値は8bitが最適
- 今後の予定
  - NTMobileに正式仕様として適用

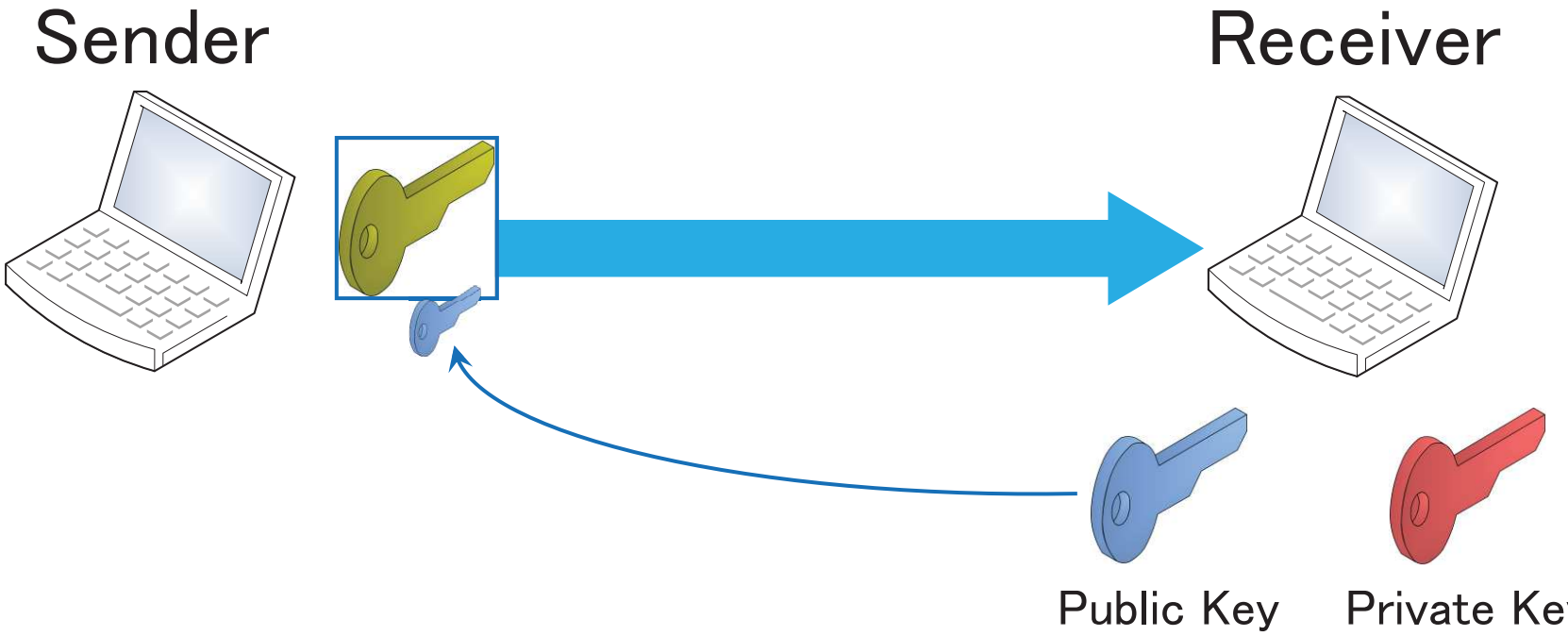
DoS攻撃対策(鍵共有時)

- 送信元の存在を確認
  - 軽い処理を実施後，重い処理を実施



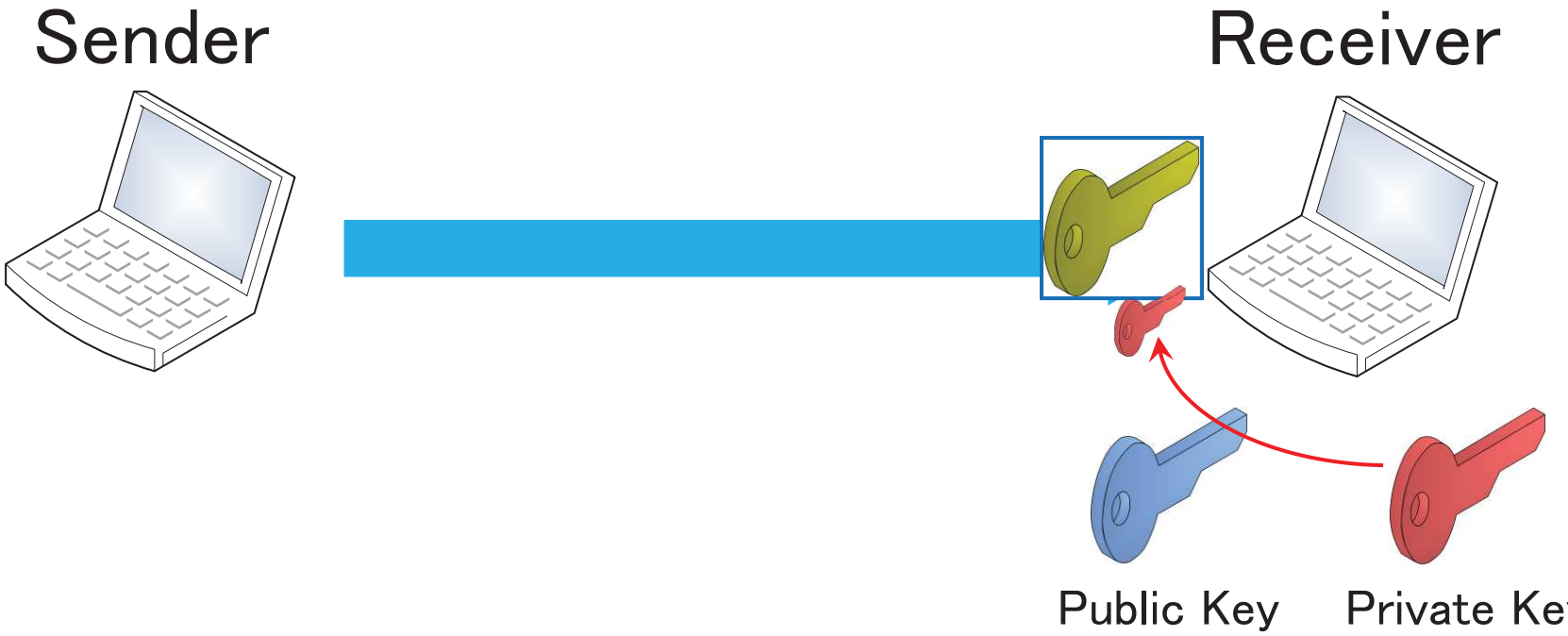
DoS攻撃対策(鍵共有時)

- 送信元の存在を確認
  - 軽い処理を実施後，重い処理を実施



DoS攻撃対策(鍵共有時)

- 送信元の存在を確認
  - 軽い処理を実施後，重い処理を実施



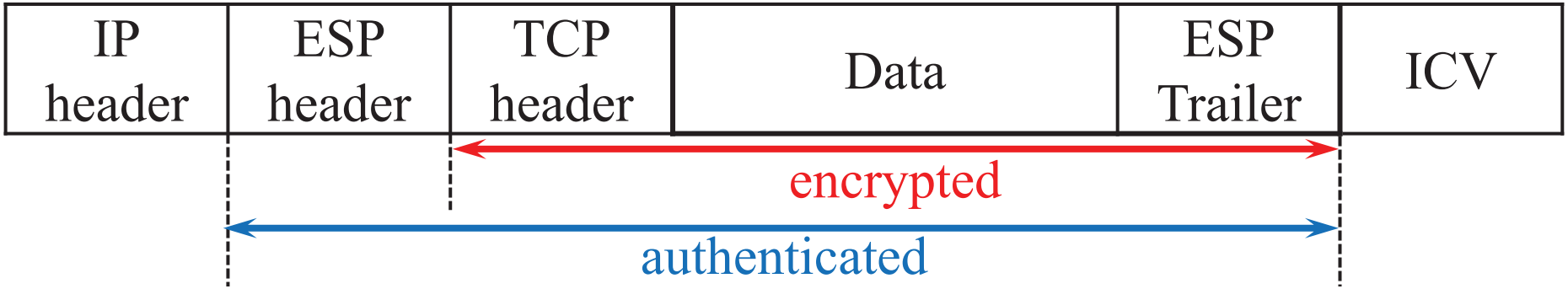
Watanabe Lab.

既存技術補足(ESP)

- ESP (Encapsulating Security Payload)
  - パケットの機密性および完全性を確保し
    送信元の認証を行うセキュリティプロトコル
    - 機密性: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
    - 完全性: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
  - 機密性の確保:パケットの暗号化
  - 完全性の確保・送信元の認証:MAC認証
    - ESPでは, MACをICV (Integrity Check Value)と呼ぶことが多い

Watanabe Lab.

ESPのパケットフォーマット

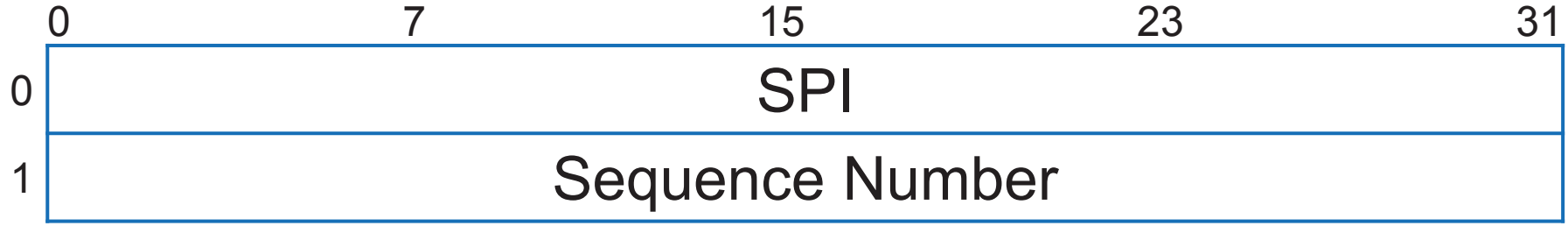


- ESP header：SPI（32bit）とシーケンス番号（32bit）
  - SPI (Security Parameter Index)：セッション識別子
- ESP Trailer
  - Padding：パディング（0～255Byte）
  - Pad Length（8bit）：Paddingの長さ（Byte単位）
  - Next Header（8bit）：Dataの先頭ヘッダのIPプロトコル番号
- ICV：認証コード

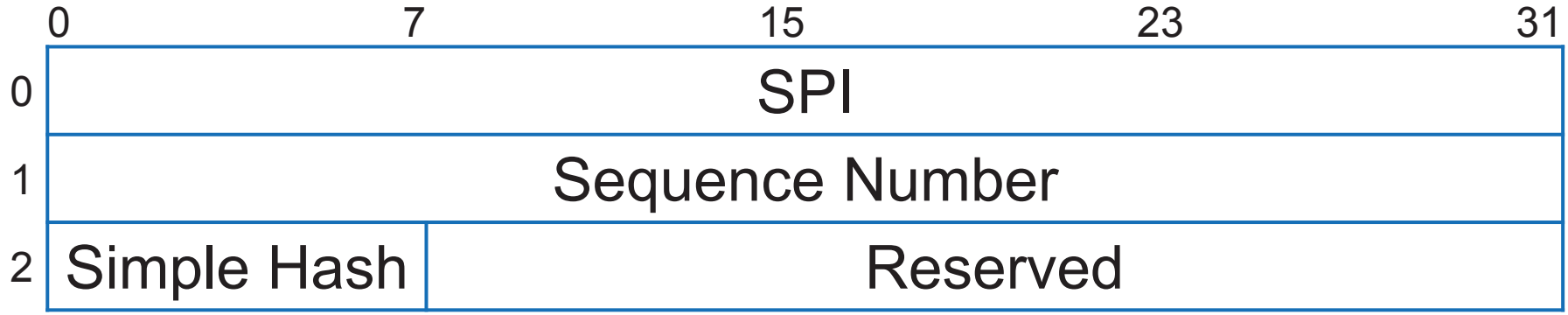
Watanabe Lab.

ESP headerの変更

- 現状のESP header



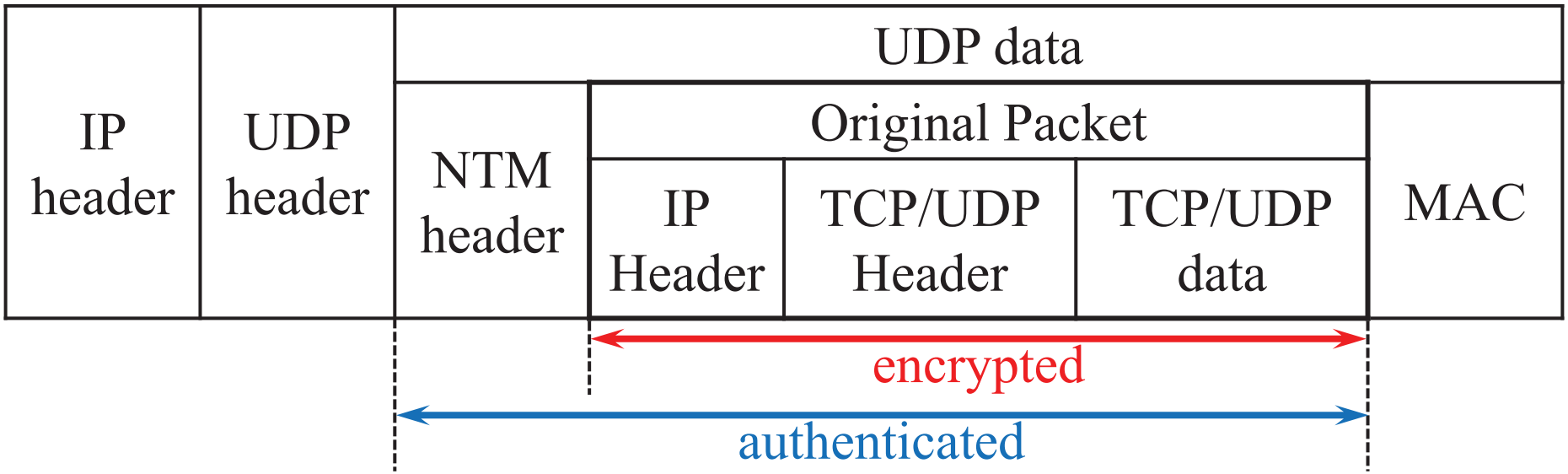
- 提案方式適用時のESP header（一例）



  - 課題：フォーマット変更により互換性がなくなる

Watanabe Lab.

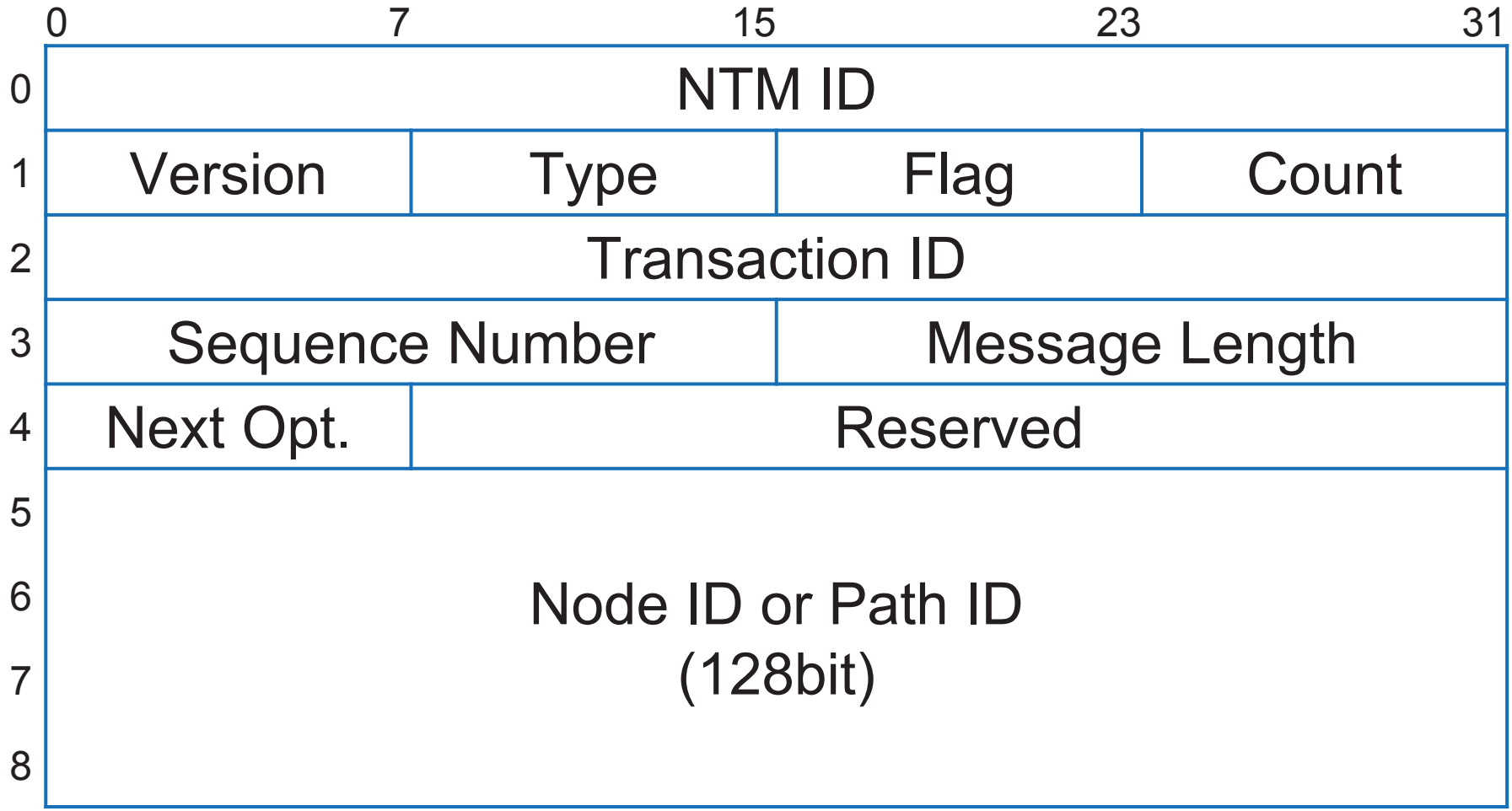
NTMobileのパケットフォーマット



- NTM header (36Byte)
  - NTMobileの通信に関わるデータを含む
    - シーケンス番号(32bit), 簡易ハッシュ値など
    - ただし, 現状のシーケンス番号は16bit
- Original Packetをカプセル化

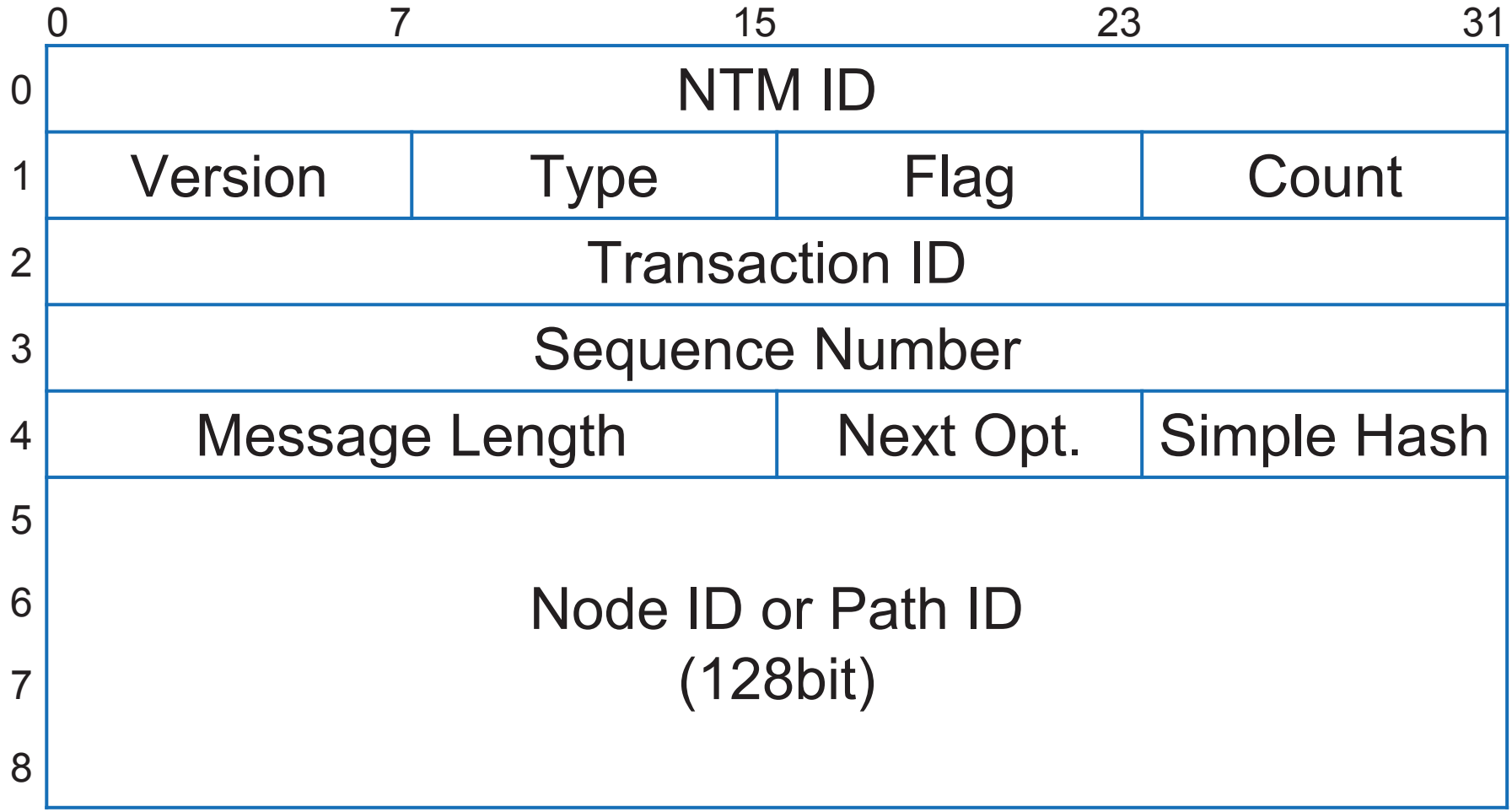
NTM headerの変更

■現状のNTM header



NTM headerの変更

- 提案方式適用時のNTM header



Watanabe Lab.

ハッシュ関数の比較

- 以下のハッシュ関数を比較
  - 除算法
  - 乗算法
  - FNV-1 32bit版
- 入力160bitのときの処理時間を測定

アルゴリズム	処理時間[μs]
除算法	0.544
乗算法	1.247
FNV-1 32bit版	0.410

  - 演算に要する時間が最も短いFNV-1 32bit版を採用

Watanabe Lab.

不正パケットの検出確率(1/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

- $\overline{P_S}, \overline{P_r}, \overline{P_m}$は以下の式で計算する
  - 各変数・定数の意味は以降に詳述

$$\overline{P_S} = 1 - \frac{1}{2^{l_h}}$$

$$\overline{P_r} = \frac{1}{2^{l_h}} \left[1 - \frac{\{(2^{l_n} - 1) - n_l\} + \{\min(s_w, n_l) - r\}}{2^{l_n}} \right]$$

$$\overline{P_m} = \frac{1}{2^{l_h}} \frac{\{(2^{l_n} - 1) - n_l\} + \{\min(s_w, n_l) - r\}}{2^{l_n}}$$

Watanabe Lab.

不正パケットの検出確率(2/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

- l_hは「簡易ハッシュ値の長さ[bit]」
  - 先述した通り$l_h = 8$

$$\overline{P_s} = 1 - \frac{1}{2^8}$$

$$\overline{P_r} = \frac{1}{2^8}\left[1 - \frac{\{(2^{ln} - 1) - n_l\} + \{\min(s_w, n_l) - r\}}{2^{ln}}\right]$$

$$\overline{P_m} = \frac{1}{2^8}\frac{\{(2^{ln} - 1) - n_l\} + \{\min(s_w, n_l) - r\}}{2^{ln}}$$

不正パケットの検出確率(3/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

- l_nは「シーケンス番号の長さ[bit]」
  - 先述した通り$l_n = 32$

$$\overline{P_s} = 1 - \frac{1}{2^8}$$

$$\overline{P_r} = \frac{1}{2^8}\left[1 - \frac{\{(2^{32} - 1) - n_l\} + \{\min(s_w, n_l) - r\}}{2^{32}}\right]$$

$$\overline{P_m} = \frac{1}{2^8}\frac{\{(2^{32} - 1) - n_l\} + \{\min(s_w, n_l) - r\}}{2^{32}}$$

Watanabe Lab.

不正パケットの検出確率(4/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

- s_Wは「リプレイ防御ウィンドウのサイズ」
  - ESPと同様とすれば$s_W = 32$

$$\overline{P_s} = 1 - \frac{1}{2^8}$$

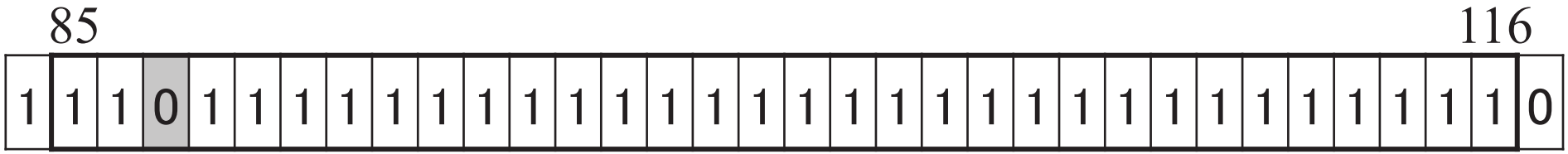
$$\overline{P_r} = \frac{1}{2^8}\left[1 - \frac{\{(2^{32} - 1) - n_l\} + \{\min(32, n_l) - r\}}{2^{32}}\right]$$

$$\overline{P_m} = \frac{1}{2^8}\frac{\{(2^{32} - 1) - n_l\} + \{\min(32, n_l) - r\}}{2^{32}}$$

不正パケットの検出確率(5/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

- n_lは「検証処理開始時の最新シーケンス番号」
- rは「リプレイ防御ウィンドウ内の受信済み個数」
  - 以下の場合, $n_l = 116$, $r = 31$



0:未受信, 1:受信済み
太枠:リプレイ防御ウィンドウ($s_W = 32$)

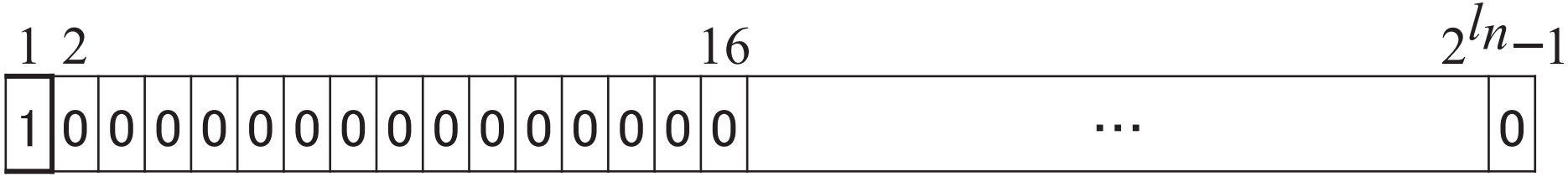
Watanabe Lab.

不正パケットの検出確率(6/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

- n_lは「検証処理開始時の最新シーケンス番号」
- rは「リプレイ防御ウィンドウ内の受信済み個数」
  - 今回のシミュレーションでは，$n_l = 1$，$r = 1$とする

    → シーケンス番号1のパケットのみを受信した状態



0:未受信，1:受信済み
太枠:リプレイ防御ウィンドウ
シーケンス番号の長さ:$l_n = 32$

Watanabe Lab.

不正パケットの検出確率(7/7)

$$E = t_S \overline{P_S} + (t_S + t_r) \overline{P_r} + (t_S + t_r + t_m) \overline{P_m}$$

■ $n_l = 1$, $r = 1$を代入すると,

$$\overline{P_s} = 1 - \frac{1}{2^8} = 9.961 \times 10^{-1}$$

$$\overline{P_r} = \frac{1}{2^8}\left[1 - \frac{\{(2^{32} - 1) - 1\} + \{\min(32, 1) - 1\}}{2^{32}}\right] = 1.819 \times 10^{-12}$$

$$\overline{P_m} = \frac{1}{2^8}\frac{\{(2^{32} - 1) - 1\} + \{\min(32, 1) - 1\}}{2^{32}} = 3.906 \times 10^{-3}$$

Watanabe Lab.

簡易ハッシュ値の長さによる比較

- 簡易ハッシュ値の長さl_hを変化させた場合
  - プログラムの制約上，最小は8bit
  - 各確率の変化は以下のようになる

l_h[bit]	t_s[μs]	$\overline{P_s}$	$\overline{P_r}$	$\overline{P_m}$
8	0.536	9.9609×10^{-1}	1.8190×10^{-12}	3.9063×10^{-3}
16	0.675	9.9998×10^{-1}	7.1054×10^{-15}	1.5259×10^{-5}
32	0.823	9.9999×10^{-1}	1.0842×10^{-19}	2.3283×10^{-10}

  - l_hが大きくなるほど，$\overline{P_s}\approx1$, $\overline{P_r}\approx0$, $\overline{P_m}\approx0$となることがわかる