

Karp-Rabin 法を用いたシグネチャ型 IDS の性能コスト評価

小林 礼明[†] 阿部 公輝[†]

シグネチャ型侵入検知システム (Intrusion Detection System, IDS) において検出処理を多段化し, 高速度を求められるステップにハードウェアを, 高精度を求められるステップにはソフトウェアを用いる一般的なアーキテクチャを提案する. 処理が単純で, 並列性を引き出しやすく, ハードウェア実装に適するパターンマッチアルゴリズム Karp-Rabin 法に着目し, そのフィルタリング精度と処理に要するコストの関係をシミュレーション実験により求める.

Cost Performance Evaluation of Signature-based IDS using Karp-Rabin Algorithm

NORIAKI KOBAYASHI[†] and KÔKI ABE[†]

An architecture of signature-based intrusion detection systems (IDSs) is proposed. In the multistage detection process of the architecture, hardware is used at upstream steps requiring high speed, and software is used at downstream steps requiring high accuracy. We focus on a pattern matching Karp-Rabin algorithm which is suitable for hardware implementation because it is simple and easy to exploit parallelism. We conduct a simulation experiment to measure the filtering accuracy and cost of multistage implementation of the algorithm.

1. はじめに

インターネットにおいて侵入検知システム (Intrusion Detection System, IDS) は不可欠なものとなっている. 年々ネットワークの通信速度が向上し, 侵入行為が増え続けるに従い, システムはより高速高精度の処理を求められるようになってきている.

ネットワーク型 IDS(NIDS) はインターネット上の不正な攻撃を識別し排除するために, パケットのペイロードに対する詳細な検査を行う. 従来このようなシステムはソフトウェアで実装されてきた¹⁾. しかしながら, 中程度の通信速度のネットワークにおいても, ソフトウェアのみでは回線速度のスループットですべてのトラフィックを処理することは不可能になってきている. IDS 処理に特化した専用ハードウェアの試みは幾つかあるが, ハードウェアのみでは性能コスト比は上がらない²⁾. FPGA によるフィルタをソフトウェアベースの NIDS の前段に配置する試みもあるが³⁾, 前段に配置するハードウェアフィルタのアルゴリズムはハードウェア向きとはいえず, 構成も一般的でない.

本研究では高速高精度のシグネチャ型 NIDS を実現

するために, 検出処理を多段化し, 高速度を求められるステップにハードウェアを, 高精度を求められるステップにはソフトウェアを用いる一般的なアーキテクチャを提案する. さらに, ハードウェアによる検出処理を多段化することにより, 性能コスト比を向上させることも考える. そのため, 処理が単純で, 並列性を引き出しやすく, ハードウェア実装に適するパターンマッチアルゴリズム Karp-Rabin 法⁴⁾に着目し, そのフィルタリング精度と処理に要するコストの関係をシミュレーション実験により求める.

以下, 第 2 節では関連研究, 第 3 節ではマッチング処理の多段化, 第 4 節では IDS の評価項目, 第 5 節では Karp-Rabin 法の説明, 第 6 節では評価実験と結果および考察を述べ, 第 7 節でまとめる.

2. 関連研究

シグネチャ型 IDS としてよく知られているものに Snort¹⁾ がある. NIDS の機能を持つソフトウェアであり, オフィシャルページでルールセットを配布している. ルールセットは Snort のマッチングの実行のために必要なファイルであり, 通信種類, ポート番号, 通信内容などの比較要素が記述されている. Snort に用意されているパターンマッチアルゴリズム⁵⁾は複数のパターンの効率的な検索が可能であり, 不要な比較処

[†] 電気通信大学 情報工学科
Department of Computer Science, The University of
Electro-Communications

理を省略してコストを削減するが、比較パターンが多くなると効率は落ちる。ソフトウェアによるIDS処理を高速化するために、複数種類のアルゴリズムによるハイブリッド型アプローチ^{6)~8)} やコンピュータクラスタなどのアプローチ^{9),10)} もあるが、スループットやコストは十分とはいえない。

高速な処理を実現するには、一般的なプロセッサによるソフトウェア制御で行うよりも、ASIC(Application Specific Integrated Circuit) や、FPGA(Field Programmable Gate Array) といった、処理に特化したハードウェア上で処理を実行できるシステムを作成することが有力な解決手段になる。しかし、従来のアルゴリズムをハードウェア上に実装する試みは、ハードウェア構成が複雑なため処理効率が上がらない。シグネチャをハッシングすることで性能向上を図るシステム²⁾ では、ハッシュ衝突の問題を克服するために構成が複雑化し、ハードウェアの能力を十分引き出しているとはいえない。

FPGA によるフィルタをソフトウェアベースのNIDSの前段に配置する試みもあるが³⁾、前段に配置するハードウェアフィルタのアルゴリズムはハードウェア向きとはいえず、構成も一般的でない。

3. マッチング処理の多段化

シグネチャ型IDSにおいて主要な処理はパターンマッチングである。高速化するネットワークと、増え続ける侵入行為によって、IDSはより高速度で高精度な処理を求められるようになってきているが、ソフトウェアのみでその処理を行おうとすると十分な速度が得られず、ハードウェアのみで処理を済ませようとすると、その構成の複雑さや、記憶すべき要素の多さから回路規模が増大し、望むような十分な処理性能が得られない。

大量のストリームに対する不正検出処理には高いスループットを必要とし、詳細な不正の判定には大きな計算量を必要とする。検知の対象となるストリームは大量の正常パケットに比較的小量の不正パケットが混入しているものとしてよいので、不正パケットを見逃すことなくストリームをフィルタリングしていけば、スループットを維持しながら、検出精度を上げていくことができると考えられる。

そこで本研究では、検出処理を分割し、検出精度は粗くても良いが、大量のストリームを高速に処理する必要のある上流ステップと、スループットは低くて良いが、精密な検出を必要とする下流ステップに分ける手法を検討する。上流ステップでは、高速な処理と不

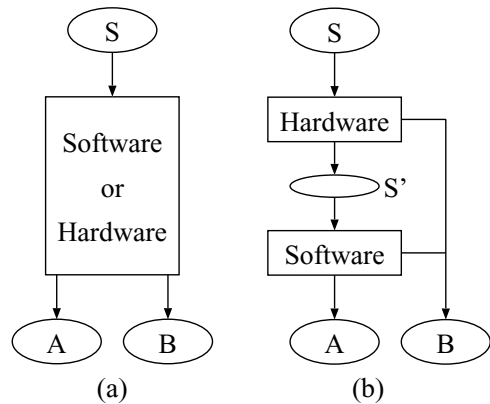


図1 従来システム (a) と提案システム (b) の比較
 Fig.1 Conventional (a) and proposed (b) systems.

正パケットを見逃さないという条件を満たす手法として、ハッシングによるパターンマッチアルゴリズムを実装したハードウェア処理を採用し、下流ステップでは、スループットは期待できないが、精密な検出が可能なソフトウェア処理を採用する。

図1に従来システム (a) と提案システム (b) を比較して示す。いずれのシステムでも、処理対象のストリーム全体を S とすると、 S を入力としたときのIDSの出力は、2つの集合 A と B から成る。 A は検知された内容、すなわち不正である文字列と不正と判断された正常な文字列から成り、 B は不正でない文字列から成る。提案システムは上流ステップと下流ステップに分割される。上流ステップの出力は S' と B から成り、 S' はすべての不正な文字列を含み、正常な文字列も含む。 B は正常な文字列のみから成る。上流ステップは精度は低いが高速度である。下流ステップは、比較的流量の小さいストリーム S' を入力とし、高精度の処理を行う。

上流ステップでは十分な性能コスト比を得るため、可能な設計選択肢を考慮する必要がある。上流ステップの構成には幾つかの方法が考えられる。図2において、まず (a) のような単体処理によるフィルタリングがある。また、(b) では上流ステップ内での処理をパイプライン化することを示している。(c) はあらかじめ定められたストリーム量になるまで処理を繰り返すようなフィルタリングを行うものである。これらの方法は、次の特徴がある。(b) の構成ではスループットの向上が期待できるが、単体処理のときよりも回路構成は増えるため、コストは (a) や (c) よりも大きい。(c) の構成では低コストで確実なフィルタリング精度を得られるが、スループットは (a)、(c) よりも低い。上流ステップには十分なスループットが要求されるので、以

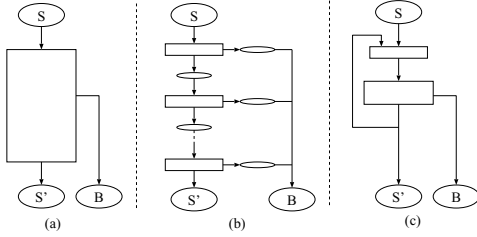


図 2 上流ステップの設計選択肢
Fig. 2 Design alternatives of upstream step.

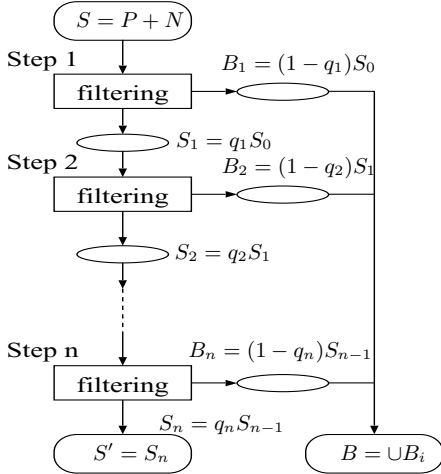


図 3 図 2(b) の構成における入出力ストリーム
Fig. 3 Streams within structure (b) in Fig. 2.

下では (b) の構成を検討する。

上流ステップの構成 (b) において、入出力ストリームが各ステップでどのようになるかを図 3 に示す。あるステップ i , $1 \leq i \leq n$, における処理後のストリーム S_i と処理前のストリーム量 S_{i-1} の比を q_i とおくと、 $S_i = q_i S_{i-1}$ であり、このステップではストリーム量は $B_i = (1 - q_i) S_{i-1}$ だけ減少する。上流ステップの出力

$$S' = S_n = q_1 q_2 \cdots q_n S_0$$

が下流ステップに与えられる。処理すべきストリーム量は

$$B = ((1 - q_1) + q_1(1 - q_2) + q_1 \cdots q_n) S_0$$

だけ減少する。

4. IDS の評価

IDS の性能を評価する項目としては、検出精度とスループットがある。

4.1 検出精度

IDS の検知結果は以下のように分類される。

- 正検知
 - True Positive (TP) : 正常なものを正常と判断する
 - True Negative (TN) : 攻撃を攻撃と判断する
- 誤検知
 - False Positive (FP) : 正常なものを攻撃と判断する
 - False Negative (FN) : 攻撃を正常なものと判断する

全ての検知結果数を $Total$, 正検知率を T , 誤検知率を F とする。

$$T = \frac{TP + TN}{Total}, F = \frac{FP + FN}{Total}, (T + F = 1)$$

上流ステップにおいて、正常な内容を攻撃と判断することは許すが、攻撃を正常なものと判断することはないと仮定する。すなわち $FN = 0$ とする。ストリーム S 中の攻撃の数を N とおくと、上流ステップでは $TP = |B|$, $TN = |N|$, $FP = |S'| - |N|$, $FN = 0$ となるので、正検知率と誤検知率は次式で与えられる。

$$T = \frac{|B| + |N|}{S}, F = \frac{|S'| - |N|}{S}$$

4.2 スループット

ステップ i , $1 \leq i \leq n$, において入力ストリーム S_{i-1} を処理するのに要する時間を t_i とする ($S_0 = S$)。ステップ i 単独のスループットを $f_i = S_{i-1}/t_i$ とする。IDS 全体のスループットを f とすると、 $f = \min \{f_i | 1 \leq i \leq n\}$ で与えられる。 $f_1 = \dots = f_n$ のとき、 f は最大になる。

ステップ i における処理時間 t_i は回路の複雑さと密接に関係する。本稿では、回路の複雑さを表すパラメータが与えられたとき、入力ストリーム S に対する上流ステップの出力 S' を測定する。

5. Karp-Rabin 法

Karp-Rabin 法は文字列のパターンマッチングを次のように行う。まず、シグネチャのハッシュ値を記憶する。対象文字列の各部分文字列に対しハッシュ値を計算し、シグネチャとのマッチングを行う。

ハッシュ値の計算の例を次式に示す。

$$\begin{aligned} 14152 &\equiv (31415 - 3 \times 10000) \times 10 + 2 \pmod{13} \\ &\equiv (7 - 3 \times 3) \times 10 + 2 \pmod{13} \\ &\equiv 8 \pmod{13} \end{aligned}$$

これは 10 進数 13 を法とする剰余をハッシュ関数とし、部分文字列 31415 のハッシュ値 7 が求まっている

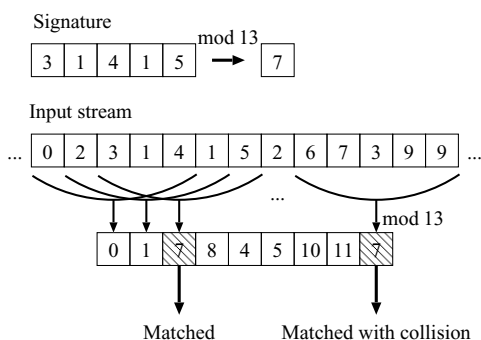


図 4 Karp-Rabin 法の実行例
Fig. 4 Illustration of Karp-Rabin algorithm.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80
(content:"GET"; msg:"HTTP Get Request");
```

図 5 Snort のルールの例
Fig. 5 A rule of Snort.

とき、文字列の位置を 1 つ進め、次の文字列 14152 のハッシュ値を計算する例である。計算式中の乗算や剰余演算は、ビットシフトとテーブルを引くことによって実行できるので、ハードウェア実装に適する。

マッチング処理の例を図 4 に示す。図において、シグネチャのハッシュ値 7 は 2 つの部分文字列でマッチしているが、一方で衝突が起きている。衝突が起きると、正常なものを攻撃と判断し誤検知 FP が生ずるが、攻撃を正常と判断する誤検知 FN は生じない。ハッシュ値の衝突頻度はハッシュ値の値域の大きさにより変わる。

6. 評価実験

上流ステップに図 2 の構成 (b) を用いるとする。不正な文字列 (シグネチャ) として Snort ルールセットを利用する。ハードウェアアルゴリズムには Karp-Rabin 法を用いる。検出精度とスループットはステップ数および各ステップのハッシュの値域に依存する。本実験では、ハッシュの値域とステップ数を変えたときの検出精度とストリーム量の変化を調べる。

6.1 実験方法

実験の流れを図 6 に示す。実験で用いる不正な文字列は、Snort が配布しているルールセットから抽出する。Snort のルールの例を図 5 に示す。このルールが表す意味は、“外部ネットワークの任意のポートから自ネットワーク上のマシンの 80 番ポートに TCP 接続で送信されたパケットの内部に GET という文字列が存在した場合に HTTP Get Request という文字列とと

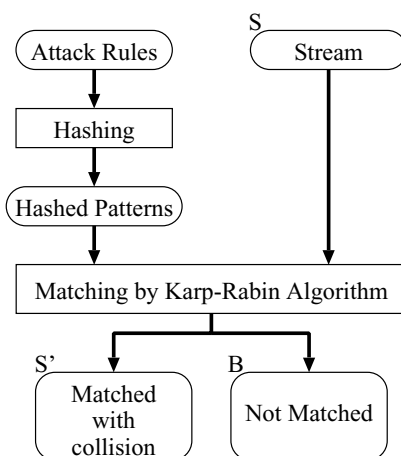


図 6 実験の流れ
Fig. 6 Experimental flow.

もに警告を発する”，という反応をするものである。通信方法のカテゴリごとにルールがまとめられている。

パターンマッチに用いられるのは **content** というオプションに記述された文字列であり、上記の例では GET である。このような文字列を 100 個抽出する。文字列の長さは 4~36 に分布している。これらの文字列のハッシュ値を求め、記憶する。Karp-Rabin 法によるパターンマッチの入力ストリームを S とする。 S の中の不正な文字列の割合を p , $0 < p < 1$, とする。

ハッシュ値の衝突を許す Karp-Rabin 法を S に適用した時の出力は、2 つの集合 S' と B から成る。 S' は FP と TN , すなわち不正な文字列と不正と判断された正常な文字列から成り、 B は TP , すなわち不正でない文字列から成る。ハッシュ関数として、素数 h を法とする剰余演算を用い、 h のビット長 $\log_2 h$ による入力ストリームの量 $|S|$ に対する S' の量 $|S'|$ の割合 $|S'|/|S|$ の変化を測定する。

6.2 実験結果と考察

入力ストリーム P の中の不正な文字列の割合 p を 0 から 1 の範囲で変化させ、素数 h のビット長を変えて $|S'|/|S|$ を測定した結果を図 7 に示す。

素数 h のビット長が大きくなるにつれ $|S'|/|S|$ の値が p に近づく。例えば、 $p = 0.1$ の場合、 $\log_2 h = 10$ のとき、 $|S'|/|S| = 0.74$ であるが、12 のときは 0.35, 14 のときは 0.15 である。また、 $p = 0.05$ の場合、 $\log_2 h = 10$ のとき、 $|S'|/|S| = 0.65$ であるが、12 のときは 0.25, 14 のときは 0.13 である。このように、あるステップのフィルタリング精度はその処理に要するコストに依存して変化する。さらにその関係は、ストリーム中の不正な

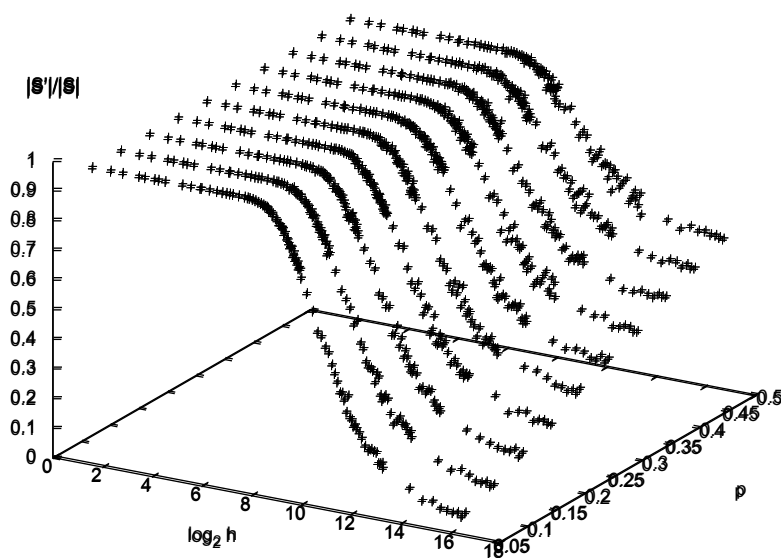


図 7 入力ストリーム S 中の不正な文字列の割合 p を 0 から 1 の範囲で変化させたときの素数 h のビット長と $|S'|/|S|$ の関係
Fig. 7 $|S'|/|S|$ as a function of p and $\log_2 h$.

パケットの混入率 p により変わる。

図 3 の構成でステップ数を変えたときの精度とストリーム量の変化を図 7 を利用して求める。図 8 と図 9 は $|S| = 400$, $|N| = 100$, すなわち不正な文字列の割合が $p = 0.25$ の入力ストリーム S が与えられたとき、ビット長 $\log_2 h$ で繰り返し処理を行ったときの精度 $T = (|B| + |N|)/S$ と上流ストリームの出力 S' を示す。例えば、ビット長が 10 のとき、単体処理では正検知率は 0.45 であるが、処理を繰り返すと 2 回目では 0.69 となり、これはビット長 11 のときの単体処理よりも精度が高い。3 回目では 0.77 に向上し、ビット長 12 の単体処理より精度が高くなる。また、ビット長が 10 のとき単体処理では上流ストリームの出力 S' は 320 であるが、処理を繰り返すと、2 回目では 220 となり、ビット長 11 のときの単体処理よりもストリーム量が少ない。また、3 回目では 190 に減少し、ビット長 12 の単体処理よりストリーム量が少ない。

図 8 において、ある程度のビット長があれば、処理を繰り返すことにより精度が向上し、出力ストリーム $|S'|$ が低減する。だが、 $\log_2 h \sim 8, 9$ ビットでは、処理を繰り返しても精度やストリーム量にほとんど変化がない。この原因を調べるために、不正な文字列の割合

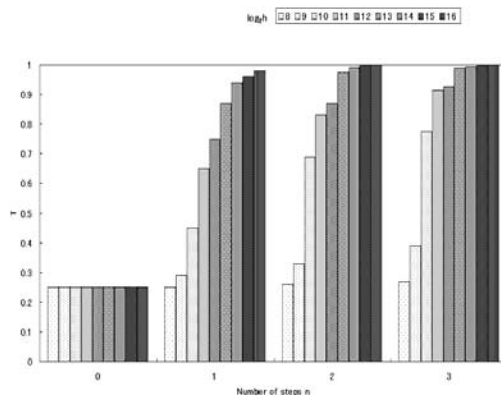


図 8 ステップ数 n による精度 T の変化
Fig. 8 Detection accuracy T depending on the number of steps n .

を $p = 0.25$ に固定し、シグネチャ数を変化させて図 7 と同様の実験を行った。図 10 にその結果を示す。図から、シグネチャ数が増えるほど、 $|S'|/|S|$ を減少させるには長いビット長が必要であることがわかる。これは、ハッシュの値域を一定とすると、シグネチャの数が大きくなるにつれ、シグネチャのハッシュ値が衝突す

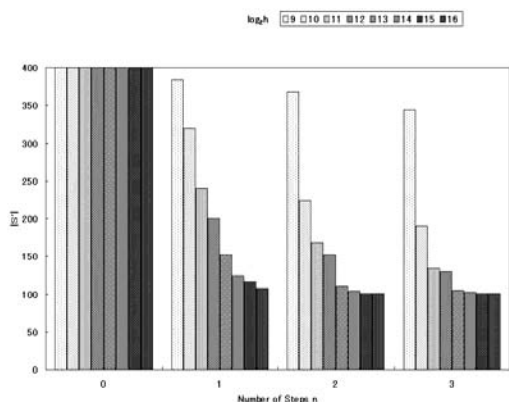


図 9 ステップ数 n によるストリーム量 $|S'|$ の変化
Fig. 9 Amount of $|S'|$ depending on the number of steps n .

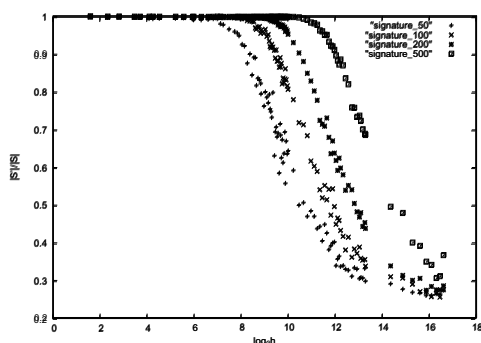


図 10 シグネチャ数による $|S'|/|S|$ の変化
Fig. 10 $|S'|/|S|$ depending on the number of signatures.

る頻度が高くなることによる。したがって、本手法においては、検出するシグネチャ数に応じてハッシュのビット長を適切に選択する必要がある。

7. おわりに

シグネチャ型 IDS において、検出処理を多段化し、高速度を求められる上流ステップにハードウェアを、高精度を求められる下流ステップにはソフトウェアを用いる一般的なアーキテクチャを提案した。さらに、ハードウェアによる検出処理を多段化する際、ハードウェア実装に適するパターンマッチアルゴリズム Karp-Rabin 法に着目し、そのフィルタリング精度と処理に要するコストの関係をシミュレーション実験により求めた。ハッシュ値のビット長は回路の複雑さを表し、回路の複雑さは処理時間と密接に関係する。Karp-Rabin 法のハードウェア実装を行い、ハッシュ値のビット長とハードウェアコストとの関係を明らかにすることは今

後の課題である。

謝辞 本研究は、一部、日本学術振興会科学研究費補助金 (基盤研究 (C)(2)18500048) による。

参考文献

- 1) Snort - the de facto standard for intrusion detection/prevention <http://www.snort.org/>
- 2) J. Botwicz, P. Buciak, and P. Sapiecha, "Building Dependable Intrusion Prevention Systems," *Proc. International Conference on Dependability of Computer Systems*, pp.135-142, 2006.
- 3) H. Song, T. Sproull, M. Attig, and J. Lockwood, "Snort offloader: a reconfigurable hardware NIDS filter," *Proc. International Conference on Field Programmable Logic and Applications*, pp.493-498, 2005.
- 4) R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM Journal of Research and Development*, pp.249-260, 1981.
- 5) M. Norton, D. Roelker, "Snort 2.0: High Performance Multi-Rule Inspection Engine," 2002.
- 6) C. J. Coit, S. Staniford, and J. McAlerney, "Towards faster pattern matching for intrusion detection or exceeding the speed of snort," *Proc. DARPA Information Survivability Conference and Exposition*, Vol.1, pp.367-373, 2001.
- 7) E. P. Markatos, S. Antonatos, M. Polychronakis, and K. G. Anagnostakis, "Exclusion-based signature matching for intrusion detection," *Proc. IASTED International Conference on Communication and Computer Network*, pp.146-152, 2002.
- 8) N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection," *Proc. IEEE Infocom*, pp.333-340, 2004.
- 9) C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful Intrusion Detection for High-speed Networks," *Proc. IEEE Symposium on Security and Privacy*, pp.285-293, 2002.
- 10) K. Watanabe, N. Tsuruoka, and R. Himeno, "Performance of Network Intrusion Detection Cluster System," *Proc. The 5th International Symposium on High Performance Computing*, pp.278-287, 2003.