An Efficient FPGA Implementation of Principle Component Analysis based Network Intrusion Detection System

Abhishek Das Sanchit Misra Sumeet Joshi[†] Joseph Zambreno[‡] Gokhan Memik Alok Choudhary

Electrical Engineering and
Computer Science Department
Northwestern University,
Evanston IL, USA

[†]Department of Electronics and
Electrical Communications
Indian Institute of Technology,
Kharagpur, India

[‡]Electrical and Computer
Engineering Department
Iowa State University
Ames, IA, USA

Abstract

Modern Network Intrsuion Detection Systems (NIDSs) use anomaly detection to capture malicious attacks. Since such connections are described by large set of dimensions, processing these huge amounts of network data becomes extremely slow. To solve this time-efficiency problem, statistical methods like Principal Component Analysis (PCA) can be used to reduce the dimensionality of the network data. In this paper, we design and implement an efficient FPGA architecture for Principal Component Analysis to be used in NIDSs. Moreover, using representative network intrusion traces, we show that our architecture correctly classifies attacks with detection rates exceeding 99.9% and false alarm rates as low as 1.95%. Our implementation on a Xilinx Virtex-II Pro FPGA platform provides a core throughput of up to 24.72 Gbps, clocking at a frequency of 96.56 MHz.

1. Introduction

Network Intrusion Detection Systems (NIDSs) are used to monitor suspicious network activity. In the wake of increasing relevance of cyber-security, they are widely used in current communication networks. NIDSs can be classified into two types: *signature detection* and *anomaly or outlier detection*. Signature detection, or misuse detection, searches for well-known patterns of attacks and intrusions by scanning for pre-classified signatures in TCP/IP packets. On the other hand, anomaly detection is used to capture behavior that deviates from the norm. These methods take as input training data to build normal network behavior models. Alarms are raised when any activity deviates from the normal model. The above models are generated using statistical analysis, data mining algorithms, genetic algorithms, etc.

Anomaly detection can detect new intrusions while misuse detection may not. However, a drawback is that anomaly detection methods suffer from false alarms. They may raise alarms for normal activity (false positives) or not sound alarms during attacks (false negatives). The need to design solid anomaly detection methods is imperative. The number of new attacks is increasing and variations of even known attacks cannot be recognized by signature detection.

In this paper, we develop a novel architecture for Principal Component Analysis (PCA) that is used as an outlier detection method. Principal component analysis is appealing since it effectively reduces the dimensionality of the data and therefore reduces the computational cost of analyzing new data. In our experiments (described in Section 4), even though each connection record has 41 features, we show that PCA can effectively achieve over 99.9% detection rate with only 7 principal components.

Reconfigurable hardware solutions are an attractive implementation choice for anomaly detection due to their inherent parallelism, pipelining characteristics, and adaptability. Hence, we implement our design on a Xilinx Virtex-II Pro FPGA platform, taking advantage of this ample design flexibility. Overall, our system is able to achieve a throughput of up to 24.72 Gbps, which can satisfy the needs of Gigabit connections. In addition to its use in network intrusion detection, PCA is a commonly used statistical method and hence our implementation can be widely utilized in various domains.

The rest of the paper is organized as follows. Section 2 contains the related work regarding hardware implementation of NIDSs. Section 3 describes the concept of PCA and illustrates in detail how it can be tailored for use in NIDS applications. The implementation details, performance and speedup reults of our FPGA architecture of PCA are shown in Section 4. Finally the paper is concluded with a brief summary in Section 5.

2. Related Work

Many reconfigurable architectures have been implemented for intrusion detection. Baker et. al was able to implement a version of the Apriori [3] algorithm using systolic arrays and also look into efficient pattern matching [2] as a signature based method. Sidhu and Prasanna also implemented a pattern matching architecture for FPGAs [11]. Attig et. al proposed a framework for rule processing on FPGAs [1]. Many packet processing architectures for FPGA have been implemented. The scope of these applications range from string matching, payload processing, packet classification, and TCP flow processing [4, 12, 14], but do not include anomaly detection.

The Karhunen Lo'eve Transform, which uses the concepts of PCA, has been mapped to FPGAs in the past [5] for use with multi-spectral imagery suitable for remote-sensing applications. However, this application does not decouple the eigenanalysis step from the main pipeline which we accelerate for network intrusion detection.

Shyu et al. [13] uses PCA on the KDD CUP 1999 data set [9]. We use a similar PCA methodology and in our tests, we verify that PCA is an effective outlier detection method for network intrusion detection. However, these studies did not consider hardware implementations of the algorithms or their applicability to hardware implementation. In summary, to the best of our knowledge, there has not been a previous principal component analysis implementation on FPGA hardware. We develop such an architecture and utilize it for network intrusion detection.

3. Principal Component Analysis

PCA is used in a variety of domains to reduce the number of dimensions in input sets without losing the "information" contained in them. At its core, PCA produces a set of principal components, which are orthonormal eigenvalue/eigenvector pairs. In other words, it projects a new set of axes which best suit the data. In our implementation, these set of axes represent the normal connection data. Outlier detection occurs by mapping live network data onto these 'normal' axes and calculating the distance from the axes. If the distance is greater than a certain threshold, then the connection is classified as an attack. This section introduces PCA and describes how it is used in outlier detection.

3.1. PCA Methodology

Anomaly detection systems typically require more data than is available at the packet level. Using preprocessing and feature extraction methods, the data available for anomaly detection is high dimensional in nature. The computational cost of processing massive amounts of data in real time is immense. Therefore applying Principal Component Analysis as a data reduction tool while retaining the important properties of the data is useful. PCA works to explain the variance-covariance structure of a set of variables through a new set of orthonormal projection values which are linear combinations of the original variables. Principal components are particular linear combinations of p random variables X_1, X_2, ... , X_p. These variables have three important properties:
1. X_1, X_2, ... , X_p are uncorrelated,
2. X_1, X_2, ... , X_p are sorted in descending order, and
3. $X_{total} = \sum_{i=0}^{p} X_i$, the total variance is equal to the sum of the individual variances.

These variables are found from eigenanalysis of the covariance or correlation matrix of the original variables X_{o1}, X_{o2}, ... , X_{op} [6, 7].

Let the original data, in this case the training data, **X** be an **n x p** data matrix of **n** observations with each observation composed of **p** fields (or dimensions) X_1, X_2,...,X_p.

Let **R** be a **p x p** correlation matrix of X_1, X_2,...,X_p. If $(\lambda_1, \mathbf{e}_1), (\lambda_2, \mathbf{e}_2), ..., (\lambda_p, \mathbf{e}_p)$ are the **p** eigenvalue-eigenvector pairs of the correlation matrix **R**, then the i^{th} principal component is

$$\begin{aligned} y_i &= \mathbf{e}_i'(\mathbf{x} - \overline{\mathbf{x}}) \\ &= e_{i1}(x_1 - \overline{x}_1) + e_{i2}(x_2 - \overline{x}_2) \\ &\quad +... + e_{ip}(x_p - \overline{x}_p), i = 1, 2, ..., p \end{aligned}$$

where
$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$,
$\mathbf{e}_i$' = e_{i1}, e_{i2},...,e_{ip} is the i^{th} eigenvector,
$\mathbf{x} = (x_1, x_2, ..., x_p)$ is the observed data along the variables X_1, X_2,...,X_p,
$\overline{\mathbf{x}} = (\overline{x}_1, \overline{x}_2, ..., \overline{x}_p)$ is the sample mean vector of the observation data.

The principal components derived from the covariance matrix are usually different from the principal components generated from the correlation matrix. When some values are much larger than others, then their corresponding eigenvalues have larger weights.

3.2. Distance Calculation

Calculating distance from a point is a fundamental operation in outlier detection techniques. Methods include nearest-neighbor, k_{th} nearest neighbor, Local Outlier Factor, etc. In general, the distance metric used is Euclidean distance. This is the primary calculation in the nearest neighbor approach. Let **x** = (x_1, x_2, ... , x_p) and **y** = (y_1, y_2, ... , y_p) be two p-dimensional observations. The Euclidean distance is defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})'(\mathbf{x} - \mathbf{y})} \tag{1}$$

In Equation 1, each feature carries the same weight in calculating the Euclidean distance. However, when features have a varied weight distribution or are measured on different scales, then the Euclidean distance is no longer adequate. The distance metric needs to be modified to reflect

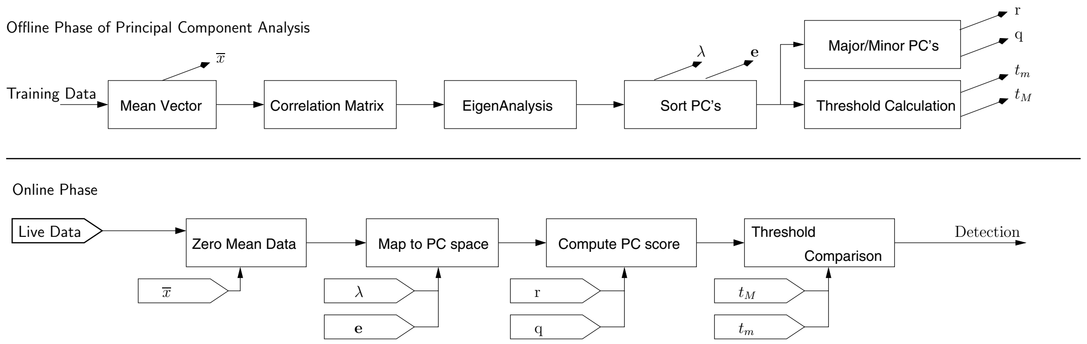

Figure 1. Principal Component Analysis for Network Intrusion Detection

the distribution and importance of each field in the data. One of these metrics is known as the Mahalanobis distance

$$d^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})' \mathbf{S}^{-1} (\mathbf{x} - \mathbf{y}) \tag{2}$$

where $\mathbf{S}^{-1}$ is the sample covariance matrix. In our work, we replaced $\mathbf{S}^{-1}$ with the correlation matrix, $\mathbf{R}^{-1}$, since many fields in the training set were measured on different scales and ranges. Using the correlation matrix more effectively represents the relationships between the data fields.

3.3. Applying PCA to Outlier Detection

In applying PCA, there are two main issues: how to interpret the set of principal components, and how to calculate the notion of distance.

First, each eigenvalue of a principal component corresponds to the relative amount of variation it encompasses. The larger the eigenvalue, the more significant its corresponding projected eigenvector. Therefore, the principal components are sorted from most to least significant. If a new data item is projected along the upper set of the significant principal components, it is likely that the data item can be classified without projecting along all the principal components.

Secondly, eigenvectors of the principal components represent axes which best suit a data sample. If the data sample is the training set of normal network connections, then those axes are considered normal. Points which lie at a far distance from these axes would exhibit abnormal behavior. Using a threshold value (t), any network connection with Mahalanobis distance greater than the threshold is considered an outlier, and hence in our case an attack.

Consider the sample principal components, y_1, y_2, ... , y_p of an observation **x** where:

$$y_i = \mathbf{e}_i'(\mathbf{x} - \overline{\mathbf{x}}), i = 1, 2, ..., p$$

The sum of squares of the partial principal component scores is equal to the principal component score:

$$\sum_{i=1}^{p} \frac{y_i^2}{\lambda_i} = \frac{y_1^2}{\lambda_1} + \frac{y_2^2}{\lambda_2} + ... + \frac{y_p^2}{\lambda_p} \tag{3}$$

equates to the Mahalanobis distance of the observation $\mathbf{X}$ from the mean of the normal sample data set [6].

3.4. PCA Framework

All anomaly detections require an offline training or learning phase whether those methods are outlier detection, statistical models, or association rule mining. Many times, the mechanisms applied in the online and offline phases are tightly coupled. Principal component analysis, however, clearly separates the offline and online detection phases. This property is an advantage for hardware implementation. Figure 1 outlines the steps involved in PCA.

In the offline phase, labeled training data is taken as input and a mean vector of the whole sample is computed. Ideally these data sets are a snapshot of activity in a real network environment. Secondly, a correlation matrix is computed from the training data. A correlation matrix normalizes all the data by calculating the standard deviation. Next, eigenanalysis is performed on the correlation matrix to extract independent orthonormal eigenvalue/eigenvector pairs. These pairs make up the set of principal components used in online analysis. Lastly, the sets of principal components are sorted by eigenvalue in descending order. The eigenvalue is a relative measure of the variance of its corresponding eigenvectors. Using PCA to extract the most significant principal components is what makes it a dimensionality reducing method because only a subset of the most important principal components are needed to classify any new data.

To increase the detection rate of PCA, we use a modified version of PCA. In addition to using the most significant principal components (q) to find intrusions, it is helpful to look for intrusions along a number of least significant components (r) as well. The most significant principal components are part of the major principal component score

(MajC) and the least significant components belong to calculating a minor principal component score (MinC). MajC is used to detect extreme deviations with large values on the original features. These observations follow the correlation structure of the sample data. However, some attacks may not follow the same correlation model. MinC is used to detect those attacks. As a result, two thresholds are needed to detect attacks. If the principal components are sorted in descending order, then q is a subset of the highest values and r is a subset of the smallest components. The MajC threshold is denoted t_M while the MinC threshold is referred to as t_m. An observation **x** is an attack if:

$$\sum_{i=1}^{q} \frac{y_i^2}{\lambda_i} > t_M \; or \sum_{i=p-r+1}^{p} \frac{y_i^2}{\lambda_i} > t_m \qquad (4)$$

The online portion takes q major principal components and r minor principal components and maps online data into the eigenspace of those principal components. There are two parallel pipelines, one for calculating the major component variability score (MajC) and one for the minor (MinC). The simulations show that adding the MinC pipeline increases the detection ability and decreases the false alarm rate of using PCA for anomaly detection. For hardware design, the most computationally expensive portion of PCA is performing eigenvector calculations and sorting. The process of calculating eigenvectors is sequential and difficult to parallelize. Fortunately, this task is part of the offline phase. We are primarily concerned with accelerating online intrusion detection using PCA. For this segment, the most important bottleneck is computing the PC score. Fortunately, this task can be parallelized as we describe in Section 4.

4. FPGA Implementation and Results

We implement our intrusion detection system on an FPGA platform, because it offers the user ample flexibility while allowing for customized designs. VHDL is used to synthesize the design, with Xilinx ISE 8.1 as the place-and-route tool. The target device is the Xilinx XC2VP30 FPGA with -6 speed grade. Our results are based upon the placed and routed design.

To examine the area and performance of PCA in real-time, we implement the online portion of the principal component score pipeline (PCSP) as shown in Figure 2. In our simulations, the input **X** to the PCSP contains 32 8-bit data fields (i.e. p = 32) for which we extracted upto 12 principal components. This workload is feasible for a real world implementation of PCA.

There are many levels of parallelism to exploit in the PSCP pipeline. They are depicted in the dashed line boxes in Figure 2. First of all, subtracting the mean vector $\overline{x}$ from the input data is done in parallel. If each data tuple has **p** fields (**X** = (x_1, x_2,...,x_p)), then **p** operations are performed in parallel. The next phase for PCA is calculating the partial component scores (parC). The element by element multiplication, using fixed point arithmetic, is performed in parallel. This operation maps the new data along each principal component axis. The first summation is accomplished with an adder tree that scales with the depth of the adder tree ($\log_2(p)$). The result is then squared and divided by the eigenvalue of the i^{th} principal component. The next step is the summation of all parC scores using another adder tree. This scales logarithmical with the number of principal components (q or r) designated. Lastly the principal component score is compared with a threshold value (t_M or t_m) determined in offline processing. The MajC and MinC pipelines have the exact same design differing only in the threshold values (t_M versus t_m) and the number of principal components used (q versus r).

Operation	# pipeline stages
a = **x** - $\overline{x}$	1
b = e_i(**x** - $\overline{x}$)	1
c = $\sum_{i=0}^{p} b_i$	$\log_2(p)$
d = c^2	1
e = $\frac{d}{\lambda}$	34
f = $\sum_{i=0}^{q} e_i$	$\log_2(z)$
outlier = f $\leq$ threshold	1
Total # pipeline stages	38 + $\log_2(p)$ + $\log_2(z)$

Table 1. Pipeline Stages Breakdown [z = max(q,r)]

Either or both of MajC and MinC pipelines detect intrusions using one correlation model from PCA. Attacks are detected on two portions of the correlation structure. As the simulations show, this method increases the detection rate and decreases the false alarm rate. From a hardware perspective, the choice of q and r affects the number of pipeline stages required. In Table 1, we show the number of pipeline stages needed for each operation in PCSP; here max(q,r) can be substituted in for z in the table.

Table 2 shows the place and route statistics for the PCA architecture. We examine the area required and throughput possible with different configurations of the PCSP. The # slices field gives the number of slices of FPGA used. The # slices increase with increasing q and r. If we use the most significant principal components to calculate distance, PCA helps us in finding the best estimate of distance with the available area.

Figure 3 shows the speedup of PCA component over an equivalent software implementation for different configuations of PSCP. The software implementation was done on a machine with 2.4 GHz AMD Opteron and 2 GByte RAM. It is clear that for a given value of q, the speedup increases as we increase r, and vice versa. This is due to the fact that

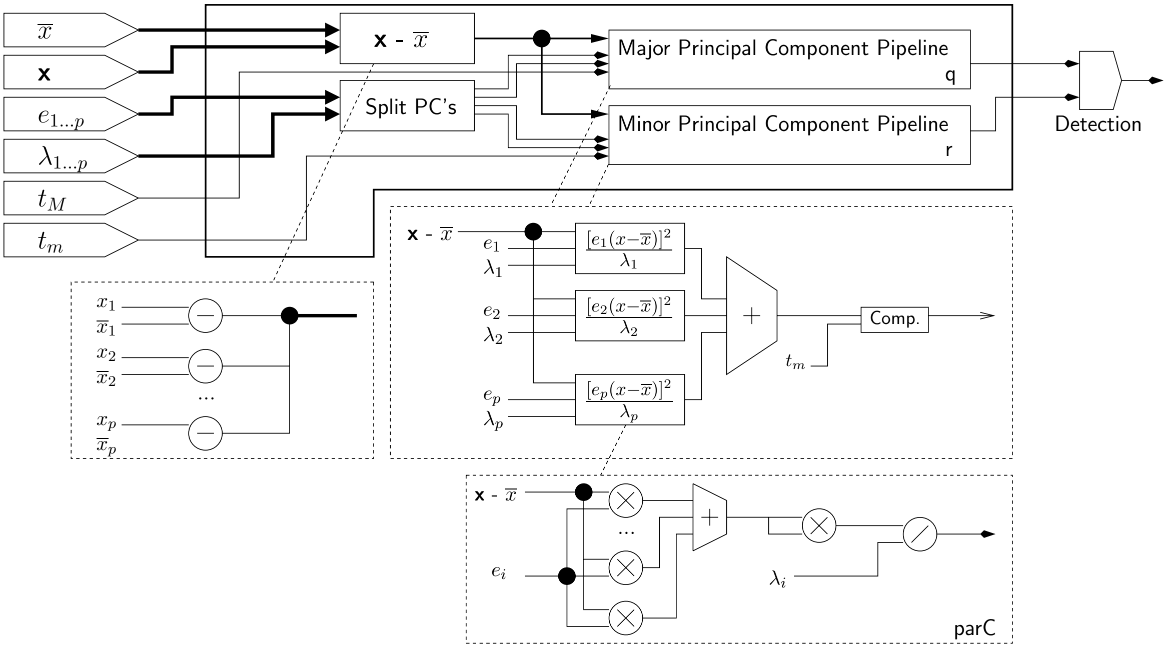

Figure 2. PCSP pipeline for FPGA

# q	# r	slices	freq (MHz)	Data Throughput (Gbps)	Pipeline Stages	Overall Latency (us)
4	0	2736	96.56	24.72	45	.4660
4	1	3002	87.62	22.43	45	.5136
4	2	3386	87.62	22.43	45	.5136
8	0	5208	87.62	22.43	46	.5250
8	1	5444	87.62	22.43	46	.5250
8	2	5823	87.62	22.43	46	.5250
8	4	6774	87.62	22.43	46	.5250

Table 2. Variation of area/latency properties for different q and r values when p (# fields) = 32

we calculate the parC value of each principal component in parallel and then use an adder tree to calculate the summation of parC values. Note that the the maximum speedup is limited by the bandwidth of the underlying hardware, which in our case is largely dependent on the I/O bandwidth of the FPGA board. Moreover, our architecture has been compared with a sequential software implementation. A parallelized implementation would result in different speedups.

To measure the effectiveness of PCA we use both training and testing data sets from the KDD Cup 1999 repository used for The Third International Knowledge Discovery and Data Mining Tools Competition [9]. Even though there has been a large amount of effort in developing efficient intrusion detection systems, there still does not exist a common benchmarking methodology to test the success of such systems. Commonly, researchers revert to the ad-hoc methods or intrusion traces [10]. Among these, KDD 1999 data set is arguably the most commonly used one (the other important trace is DARPA 1998, which is a predecessor of KDD 1999) [8]. Therefore, we focused on the KDD 1999 dataset in our work.

Table 3 shows that principal component analysis detects a high percentage of attacks with a low false alarm rate. The testing data sets in this table each have between 100,000 to 125,000 network connections randomly extracted from the testing data. We only use 32 of the 41 features in our experiments. The remainder of the features were either symbolic or contained only zero values. It is clear from the results that adding the MinC pipeline for network connection boosts the detection rate and decreases the false alarm rate. In the case of (q, r)=(3, 5), the average detection rate in Table 3 is 99.92% and the average false alarm rate decreases to 2.13%. For some input sets, (3, 5) performs better than (5, 5). This is due to the random distribution of attacks and normal connections. By including more components, we may actually miss an attack, because different configurations will have different threshold values. Some normal attacks may seem like attacks and vice versa. However in

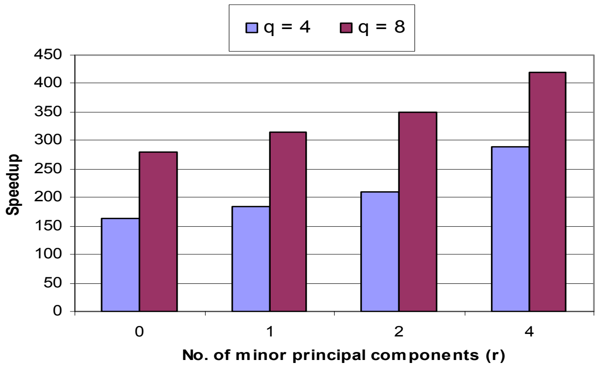

Figure 3. PCA hardware speedups for p (# fields) = 32

Training File	q (Major)	r (Minor)	Detection	False Alarm
kddc25	3	–	75.56%	10.27%
	3	4	98.19%	5.22%
	3	5	99.99%	2.25%
	5	–	91.94%	10.52%
	5	4	97.81%	5.22%
	5	5	99.95%	2.28%
	7	–	99.91%	9.55%
kddc70	3	–	62.60%	12.50%
	3	4	99.98%	3.28%
	3	5	99.90%	2.12%
	5	–	83.23%	14.23%
	5	4	98.50%	6.39%
	5	5	99.96%	2.91%
	7	–	98.74%	5.21%
kddc55	3	–	92.89%	12.11%
	3	4	99.50%	4.18%
	3	5	99.86%	2.01%
	5	–	96.21%	11.21%
	5	4	98.19%	5.17%
	5	5	99.96%	1.95%
	7	–	99.93%	10.50%

Table 3. Detection and False Alarm Rates utilizing MinC and MajC pipelines

general we see that increasing q or r increases the detection rate.

5. Conclusions

In this paper, we have designed a hardware implementation of Principal Component Analysis (PCA) module used in a Network Intrusion Detection System. PCA is particularly helpful in anomaly detection since it can reduce the data dimensionality into a smaller set of independent variables. Our FPGA implementation used hardware parallelism and extensive pipelining, and can detect over 99.9% of the network attacks with false alarm rates as low as 1.95% for KDD 1999 cup data sets. The results also show that using our hardware architecture the system can achieve a link speed of up to 24.72 Gbps and a clock frequency of 96.56 MHz.

References

[1] M. E. Attig and J. Lockwood. A framework for rule processing in reconfigurable network systems. In *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, Napa, CA, Apr. 2005.

[2] Z. K. Baker and V. K. Prasanna. Time and Area Efficient Pattern Matching on FPGAs. In *The Twelfth Annual ACM International Symposium on Field-Programmable Gate Arrays (FPGA '04)*, 2004.

[3] Z. K. Baker and V. K. Prasanna. Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs. In *Proceedings of the Thirteenth Annual IEEE Sym. on Field Programmable Custom Computing Machines 2005*, 2005.

[4] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. W. Lockwood. Deep packet inspection using parallel bloom filters. In *Symp. on High Performance Interconnects*, August 2003.

[5] M. Fleury, B. Self, and A. C. Downton. A fine-grained parallel pipelined karhunen-loeve transform. In *17th International Parallel and Distributed Processing Symposium*, Nice, France, April 2003.

[6] J. D. Jobson. *Applied Multivariate Data Analysis, Volume II: Categorical and Multivariate Methods*. Springer-Verlag, NY, 1992.

[7] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, NY, 2002.

[8] Jungsuk Song and Hiroki Takakura and yasuo Okabe. A proposal of new benchmark data to evaluate mining algorithms for intrusion detection. In *23rd Asia Pacific Advanced Networking Meeting*, 2007.

[9] KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, August 1999.

[10] Nicholas Athanasiades and Randal Abler and John Levine and Henry Owen and George Riley. Intrusion detection testing and benchmarking methodologies. In *IEEE International Information Assurance Workshop*, 2003.

[11] R. Sidhu and V. Prasanna. Fast regular expression matching using fpgas. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2001.

[12] D. V. Schuehler, J. Moscola, and J. W. Lockwood. Architecture for a hardware-based, tcp/ip content-processing system. In *IEEE Micro*, January 2004.

[13] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. In *IEEE Foundations and New Directions of Data Mining Workshop*, pages 172–179, November 2003.

[14] H. Song and J. W. Lockwood. Efficient packet classification for network intrusion detection using fpga. In *Intl. Symp. on Field-Programmable Gate Arrays*, February 2005.