Hardware Accelerator for Low-Latency Privacy Preserving Mechanism

Junichi Sawada and Hiroaki Nishi
Graduate School of Science and Technology, Keio University
Kanagawa, Japan
Email: sawada@west.sd.keio.ac.jp, west@sd.keio.ac.jp

***Abstract*—With the recent growth in the quantity and value of data, data holders have come to realize the importance of being able to utilize information that is otherwise abandoned or concealed. In this situation, they face the difficulty of publishing data without revealing private information. Two of the methods used to protect private information when publishing data are privacy-preserving methods based on constraints known as k-anonymity and l-diversity. These methods enable the utilization of published data while preserving privacy, but incur a large computational cost. We solve this problem using a hardware architecture composed of Ternary Content Addressable Memory (TCAM), which can significantly accelerate the privacy preservation process. k-anonymity and l-diversity have not been studied in any significant way for efficient hardware implementation. Thus, this will be the first of its kind. An evaluation proves that an implementation of the proposed architecture on a reconfigurable device performs approximately 10-50 times faster than a RAM-based architecture.**

Keywords-hardware; reconfigurable device; privacy-preserving data publishing.

I. Introduction

Recently, the spread of Web services such as social networking services, blogs, and Internet shopping has emphasized the importance of users' information on Web servers and databases. Ubiquitous devices, sensor networks, and RFIDs will accelerate this situation. These types of applications generate information, including trends, which is valuable for service providers, social researches, and marketing. In this situation, data holders intend to share and utilize information that is otherwise abandoned or concealed.

In this situation, data holders face the difficulty of publishing data without revealing private information. Beyond the current methods of protecting against external cyber attacks, new methods of protecting private information when publishing data are needed. One of these is a privacy-preserving method based on a statistically proved constraint such as k-anonymity [1] or l-diversity [2] that enables data users to utilize published data under the constraint of preserving privacy. However, this method has a high processing cost, which makes it difficult to process high-throughput data streams such as the output of a database or network traffic that has been kept generated without pausing.

One of promising approaches to improve the performance is hardware implementation. We propose a TCAM-based hardware architecture for accelerating k-anonymity and l-diversity methods. These methods have not been studied in any significant way for efficient hardware implementation. Thus, this will be the first of its kind.

II. Related Work

Techniques to extract useful information without revealing privacy have been proposed for privacy-preserving data mining (PPDM) [3]. PPDM extracts useful information such as statistics and associations from more than one database with their secrecy preserved. In particular, the protection of private information when publishing data is called privacy-preserving data publishing (PPDP) [4], which is different from PPDM because it does not involve data mining. Two PPDP techniques are methods based on constraints, known as k-anonymity [1] and l-diversity [2], which are achieved by generalizing and suppressing "unique" data.

In recent years, many methods, especially for k-anonymity, have been studied. The main focus of some works is on privacy-preserving publishing of not static data, but dynamic data set, where new data can be added [5][6]. In this scenario, mainly two problems emerge; One is that the republication of the entire data set is needed whenever new data are added, and the other is the malicious inference available by analyzing the multiple versions of published data sets. To solve these problems, incremental update methods, which efficiently insert new data into the current data set without making it vulnerable, were proposed.

A clustering-based method based on k-anonymity for data streams was proposed with an eye on applications that need continuous privacy-preserving data publishing such as the publishing of telephone/network service records for network-traffic analysis, a search engine publishing a query log for online Web mining, and a stock exchange publishing its transactions [7]. Our approach is not based on clustering, and we focus on maintaining data in input order, which is important for some applications. It is different from those researches. Furthermore, our approach is based on hardware acceleration. The parallelism of our architecture efficiently accelerates the privacy preservation process.

The calculations of k-anonymity and l-diversity generally require a large cost. It has been shown that an optimal calculation of k-anonymity, which means results with the minimum information loss, is NP-hard [8]. The calculation of k-anonymity or l-diversity can be completed by repeatedly comparing each record against every other record as

 ISBN: 978-1-61208-217-2

an association-rule mining. This calculation requires a time complexity of $O(n^2)$ in the worst case for k-anonymity and in all cases for l-diversity. To improve performance, a promising approach is to exploit advanced hardware. This approach includes an implementation with network processors [9], exploitation of GPGPU or GPUTeraSort [10], improved performance of join with Cell/B.E. [11], and the implementation of special purpose hardware for stream data operations with FPGA [12][13][14][15].

The k-anonymity and l-diversity methods have not been studied in any significant way for efficient hardware implementation. Researches on the hardware implementation of an association-rule mining algorithm have been published [16][17][18]. These researches may be efficient for the k-anonymity and l-diversity methods in finding infrequent records that do not satisfy k or l. However, the requirements are different for k-anonymity or l-diversity and association-rule mining, as follows: 1. The comparison is computed for a whole record. It is not necessary to calculate the combination of elements in a record. 2. An implementation of generalization is needed. 3. Association-rule mining finds frequent records, whereas infrequent records are needed in the calculation of k and l. Moreover, association-rule mining finds a rule, not a record itself. Thus, after finding rules, another process may be needed to find records that match the extracted rules. Therefore, another implementation optimized for the privacy-preserving algorithm would be more effective than a hardware implementation of the association-rule mining algorithm.

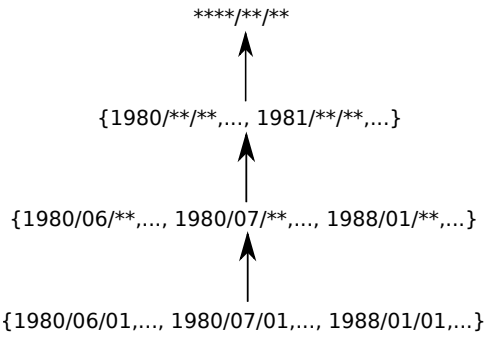

Figure 1. Birth date domain generalization hierarchy

Table I
DATA TABLE WITH SENSITIVE ATTRIBUTE "PROBLEM"

Gender	Birth	ZIP	Problem
Male	1963	02150	short breath
Male	1960	02140	chest pain
Male	1964	02138	chest pain
Male	1964	02138	obesity
Male	1964	02138	short breath

Table II
2-ANONYMIZED TABLE WITH SENSITIVE ATTRIBUTE "PROBLEM"

Gender	Birth	ZIP	Problem
Male	196*	021**	short breath
Male	196*	021**	chest pain
Male	1964	02138	chest pain
Male	1964	02138	obesity
Male	1964	02138	short breath

III. PRIVACY PROTECTION MODEL

k-anonymity and l-diversity are satisfied by using a generalization that replaces a value with a less specific, more general value, or masks a part of the value with "*." The special terms used in this paper are defined according to [1][2] as follows:

Data Table: A row is termed a tuple and a column is termed a field. Each field is said to be an attribute, which indicates the meaning of values.

Attribute: Attributes that can uniquely identify individuals such as names are termed explicit identifiers. Other attributes that in combination can uniquely identify individuals such as their birth date, ZIP, and gender are termed quasi identifiers.

Sensitive Attribute: Attributes that are not termed quasi identifiers and should not be associated with an individual, for example "diagnosis" in the case of medical data, are termed sensitive attributes. Other attributes, which are possibly the same as the quasi identifiers, are termed non-sensitive attributes and will be generalized. When the values of non-sensitive attributes are the same or have been generalized to the same values, the set of tuples is termed a *q*-block*.

Domain Generalization Hierarchy (DGH): A Domain Generalization Hierarchy (DGH) refers to a hierarchy that indicates how and how many times an attribute is generalized. For example, Figure 1 shows the DGH of the attribute "birth date."

A. k-anonymity

A table T satisfies k-anonymity if each sequence of non-sensitive values in T appears with at least k occurrences.

Table I with the sensitive attribute "problem" is generalized into Table II where $k = 2$, as an example. Because there are at least 2 of the same tuples for each tuple in Table II, an adversary cannot distinguish one tuple from another in any *q*-block*. k-anonymity guarantees that a tuple cannot be distinguished from at least $k-1$ other tuples in the table.

However, k-anonymity may leak private information in a few cases, as mentioned in [2]. In a case where the sensitive values are all the same in a *q*-block*, an adversary can infer the sensitive value even if he cannot distinguish the tuple. In another case where an adversary has a strong background knowledge about the sensitive values, he may be able to infer non-sensitive values from the sensitive values. These vulnerabilities are both caused by a lack of diversity in the sensitive values. To solve this problem, l-diversity [2] has been proposed, as stated below.

Table III
GENERALIZED VALUE EXPRESSION IN TCAM

Value	TCAM	
	Data	Mask
0011****	00110000	00001111
00111***	00111010	00000111

Table IV
GENERALIZATION PROCESS IN TCAM

Value		TCAM	
Attribute 1	Attribute 2	Data	Mask
0000	0000	0000_0000	0000_0000
	↓generalize		
000*	000*	0000_0000	0001_0001
	↓generalize		
00**	00**	0000_0000	0011_0011

B. l-diversity

Table T satisfies l-diversity if every *q*-block* has at least l different values for a sensitive attribute.

Assume that the ith most frequent sensitive value appears r_i times and n types of sensitive values appear in a *q*-block*; l-diversity is defined by equation 1.

$$r_1 \leq r_l + r_{l+1} + ... + r_n \tag{1}$$

IV. PROPOSED ARCHITECTURE

A. TCAM

A hardware implementation of the k-anonymity and l-diversity method requires the following things:

- A fast search function for infrequent tuples that do not satisfy k-anonymity or l-diversity
- Implementation of the generalization, which means a search function compatible with "*," i.e., a wild card

These requirements can be achieved using Ternary Content Addressable Memory (TCAM). CAM is a memory that receives data as input and outputs the locations where the associated contents are stored. Comparison logic for each cell enables a search operation to be completed in a single memory access. By using CAM, it becomes clear whether or not the required privacy level is satisfied in CAM with the complexity $O(n)$. Additionally, TCAM has a mask circuit that allows a third matching state of "Don't care" for each cell for various-length matching. The generalization can be processed with this circuit as shown in Table III.

By shifting mask bits, data can be generalized from the bottom bit-by-bit, as shown in Table IV.

The calculation of k-anonymity is completed by counting the number of tuples that are the same as the one specified in a search. TCAM can complete the calculation in a single cycle by searching for the tuple. Unfortunately, the calculation of l-diversity requires a process that is not as simple as that for k-anonymity because a calculation of the frequencies of the values of sensitive attributes in a *q*-block* is required. Based on the l-diversity principle expressed by equation 1, if the probability of the occurrence of a value for a sensitive attribute in a *q*-block* is less than $1/l$, a tuple that has that value has absolutely satisfied l-diversity. Thus, the probability of the occurrence of a value for a sensitive attribute in a *q*-block* can be written as $p(s|q*)$, where s denotes the sensitive value, and if the probability satisfies equation 2, the tuple is said to satisfy l-diversity.

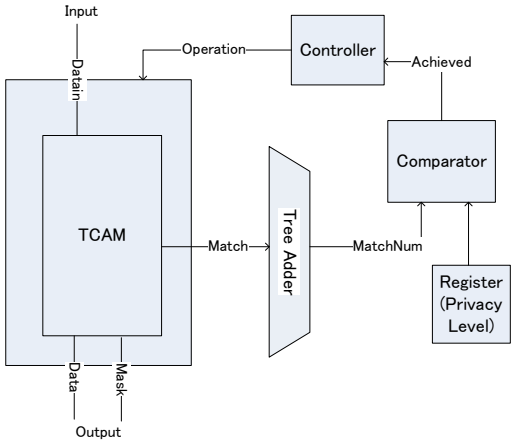

Figure 2. TCAM-based architecture for k-anonymity and l-diversity

$$p(s|q*) \leq \frac{1}{l} \tag{2}$$

Our approach to calculate the probability is to utilize variable-length matching just as in the generalization, which allows the calculation to be completed in two phases. In the first phase, a search operation is executed just as in the calculation of k-anonymity, and the total number of tuples that are the same as the one specified in the search is counted. The result of the first phase shows how many times a value for a sensitive attribute of the current tuple occurs in the *q*-block*. In the second phase, unlike the first phase, a search operation is executed with the values for the sensitive attribute masked. The result of the second phase shows the total number of tuples in a *q*-block* that include the tuple currently being processed. The probability $p(s|q*)$ of the tuple can be calculated using the two values obtained in these two phases.

B. Whole Architecture

Figure 2 shows the hardware architecture proposed in this paper. The TCAM has a match line for each entry and outputs are connected to the tree adder, which counts up the total number of asserted match lines. The comparator compares the input with the previously configured privacy level, and the output indicates whether or not the privacy

level is satisfied. The controller manages the write and read addresses, and shifts the generalization process from the current tuple to the next tuple after completing the generalization, which is necessary if the privacy level of the current tuple is not satisfied. Finally, all of the sets of data and mask values are output when all of the tuples in TCAM satisfy the privacy level. The algorithm is executed as stated below.

1) Input data into the TCAM
2) Search the TCAM for a tuple
3) Generalize the tuple if its privacy level is not satisfied
4) Search for the next tuple
5) Repeat step 3 and 4 until all of the tuples in the TCAM satisfy the required privacy level
6) Output all sets of data and mask values in the TCAM and go back to step 1

The details of this algorithm are described in the following.

Algorithm Calculation of k and l

```
1:  loop
2:    /*Input*/
3:    while TCAM is not full do
4:      write_tuple_to_TCAM;
5:    end while
6:    /*Calculation of k and l*/
7:    repeat
8:      flag ← 0;
9:      for each tuple in TCAM do
10:       search_for_tuple;
11:       if k or l is not achieved then
12:         generalize_tuple;
13:         flag ← 1;
14:       end if
15:     end for
16:   until flag is 0
17:   /*Output*/
18:   while TCAM is not empty do
19:     read_tuple_from_TCAM;
20:     result ← data | mask;
21:   end while
22: end loop
```

Because it is important to ensure the levels of tuples in the generalization hierarchy are as same as possible in computing the algorithm, the search operation at each tuple is executed only once per loop, and the loop is repeated until all of the tuples in the TCAM satisfy the required privacy level. In the output process, if the output is only one tuple, the whole transaction, including tuples that have already been output, may not satisfy l-diversity. This is why the proposed architecture exchanges tuples perfectly. In this case, because it is the same to process a divided transaction one-by-one, the architecture can work in parallel.

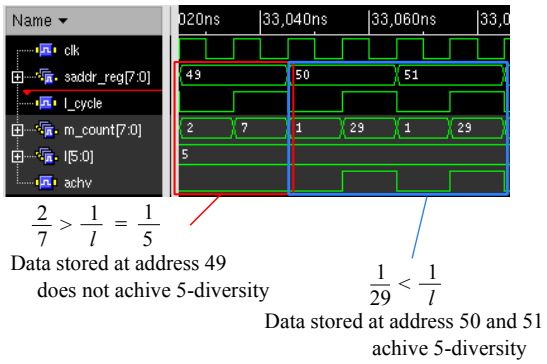

Figure 3. Observed waveforms

V. RESULTS

A. Throughput

The proposed architecture is implemented on a Xilinx Virtex5 FPGA board (XC5VLX330T) using the tool Xilinx ISE 12.3. Because the TCAM cannot be implemented using Block RAM and requires a large hardware cost, the TCAM IP core provided by Xilinx is implemented to improve resource utilization. Additionally, although the TCAM needs to output data and mask values in the proposed architecture, the TCAM IP core has no output ports for stored values. To solve this problem, the proposed architecture is emulated by storing data and mask values into not only TCAM but also Block RAMs. In the case of the 256x256 TCAM, which is the maximum entry size on XC5VLX330T, hardware usage is 45,691 LUT FF pairs and the maximum frequency is approximately 70 MHz. Figure 3 shows a capture of a SimVision waveform window. The signals described in this figure are as follows: 1. saddr_reg indicates a memory address where a tuple currently being processed is stored. 2. l_cycle indicates the two phases of the calculation of l-diversity described in IV-A. 3. m_count indicates the output of the tree adder, namely the total number of matched tuples. 4. l indicates the required l. 5. achv indicates whether the current tuple achieves l-diversity.

The throughput evaluation is performed by processing Internet traffic, specifically browsing history obtained at our laboratory to simulate a trend survey on the Web. The destination IP address is the quasi identifier and used for the non-sensitive attribute, while the Web title is the sensitive attribute. The destination IP address is processed as 32 bit data, and generalized one bit by one bit, up to 32 times. Table V shows the data layout of the data set used for this evaluation.

Figure 4 gives a throughput comparison between the proposed architecture and RAM-based architecture where the search operation is serialized. The comparison is performed with l-diversity, which requires a larger processing cost than k-anonymity.

Because the throughput is calculated by assuming that a

Table V
DATA LAYOUT OF THE DATA SET USED FOR THE EVALUATION

Destination IP address (actually 32 bit data)	Web title
012.XXX.XXX.XXX	Google
012.XXX.XXX.XXX	Facebook
123.XXX.XXX.XXX	Google
234.XXX.XXX.XXX	FIFA.com

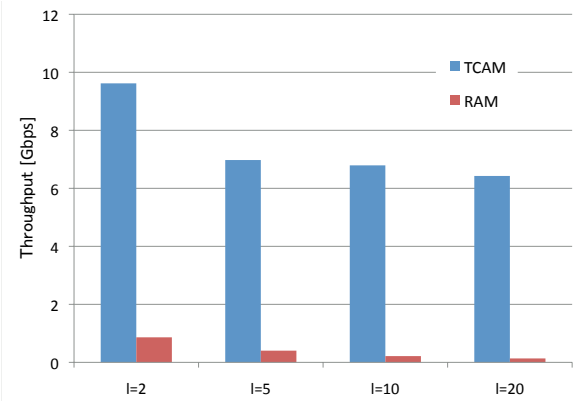

Figure 4. Performance comparison against RAM-based architecture

tuple corresponds to a packet, and the average packet size is 1,000 bytes, the architecture can process Internet traffic, in this case the browsing history in a network of at least 1 Gbps. In the case of another environment, more evaluations are needed.

In this evaluation, both the TCAM-based architecture and the RAM-based architecture process the exact same dataset. Thus, the experimental result shown in Figure 4 means that the TCAM-based architecture performs approximately 10-50 times faster than the RAM-based architecture. The CAM takes advantage of its specialty with the increase of l, as shown in the throughput differences between the two architectures, because the number of processes increases. Moreover, compared to software implementation (C++, single-threaded, 3.0 GHz Quad-Core Xeon CPUx2, 8GB DDR3) of the same algorithm, it achieves approximately 400-900 times higher throughput, as shown in Figure 5. This evaluation is also performed by processing the same dataset in the same way.

B. Information Loss

A generalized table typically has less useful information. In order to evaluate the information loss, the theoretic metric information loss (IL) of data table T, written $IL(T)$, is defined by referring to $Prec$ in [1]. Let $t \in T = \{t_1, ..., t_N\}$ be a tuple, $QI_T = \{A_1, ..., A_{N_A}\}$ be the quasi identifier, DGH be a height of a generalization hierarchy, and h be a height of a generalized value in a generalization hierarchy.

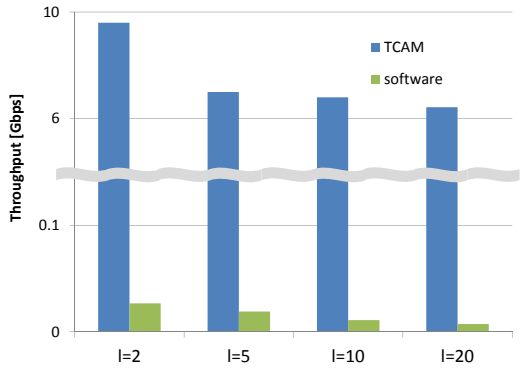

Figure 5. Performance comparison against software implementation

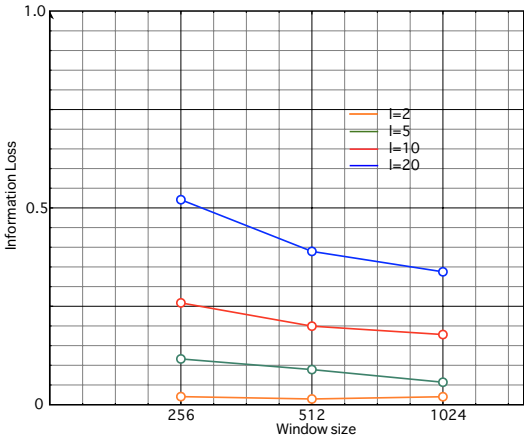

Figure 6. Trade-off between information loss and window size

Then, the equation is defined as 3.

$$IL(T) = \frac{\sum_{t_j \in T} \sum_{A_i \in QI_T} \frac{h}{|DGH_{A_i}|}}{|T| \cdot |QI_T|} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{N_A} \frac{h}{|DGH_{A_i}|}}{N \cdot N_A} \quad (3)$$

IL indicates how deep a value has been generalized with a value from 0 to 1.

To process data stream, it has to be divided in windows and processed one-by-one. Because it is difficult to satisfy a privacy level in a small window size, a trade-off exists between the information loss and a window size as shown in the graph in Figure 6.

Focusing on the utility of published information, smaller k or l would be chosen as the privacy level because it will result in smaller IL and useful information. When l is small, the increase in IL caused by the window-size constraint is also small as shown in Figure 6. In that case, the window size can be small without increasing IL, and that results in low hardware cost.

 ISBN: 978-1-61208-217-2

VI. Conclusion

A hardware architecture for the privacy-preserving algorithm based on constraints known as k-anonymity and l-diversity was proposed. The work was based on the TCAM, which enables a fast search operation. A generalization process, which is necessary for the calculation of the algorithm, was also efficiently enabled with variable-length matching. Overall, the FPGA implementation of the proposed architecture performed approximately 10-50 times faster than a RAM-based architecture where the search operation was serialized.

References

[1] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 571–588, 2002.

[2] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, March 2007.

[3] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining," *Journal of Cryptology*, vol. 15, pp. 177–206, 2002.

[4] B.-C. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala, "Privacy-Preserving Data Publishing," *Foundations and Trends in databases*, vol. 2, pp. 1–167, January 2009.

[5] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li, "Secure anonymization for incremental datasets," in *Secure Data Management*, ser. Lecture Notes in Computer Science, 2006, vol. 4165, pp. 48–63.

[6] X. Xiao and Y. Tao, "M-invariance: towards privacy preserving re-publication of dynamic datasets," in *Proceedings of the 2007 international conference on Management of data*, ser. SIGMOD '07. ACM, 2007, pp. 689–700.

[7] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia, "Continuous privacy preserving publishing of data streams," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT '09. ACM, 2009, pp. 648–659.

[8] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the 23rd SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ser. PODS '04. ACM, 2004, pp. 223–228.

[9] B. Gold, A. Ailamaki, L. Huston, and B. Falsafi, "Accelerating database operators using a network processor," in *Proceedings of the 1st international workshop on Data management on new hardware*, ser. DaMoN '05. ACM, 2005.

[10] N. Govindaraju, J. Gray, R. Kumar, and D. Manocha, "GPUTeraSort: high performance graphics co-processor sorting for large database management," in *Proceedings of the 2006 international conference on Management of data*, ser. SIGMOD '06. ACM, 2006, pp. 325–336.

[11] B. Gedik, P. S. Yu, and R. R. Bordawekar, "Executing stream joins on the cell processor," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB '07. VLDB Endowment, 2007, pp. 363–374.

[12] R. Mueller, J. Teubner, and G. Alonso, "Streams on wires: a query compiler for FPGAs," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 229–240, 2009.

[13] ——, "Data processing on FPGAs," *Proc. VLDB Endow.*, vol. 2, pp. 910–921, August 2009.

[14] L. Woods, J. Teubner, and G. Alonso, "Complex Event Detection at Wire Speed with FPGAs," *Proc. VLDB Endow.*, vol. 3, pp. 660–669, September 2010.

[15] J. Teubner and R. Mueller, "How soccer players would do stream joins," in *Proceedings of the 2011 international conference on Management of data*, ser. SIGMOD '11. ACM, 2011, pp. 625–636.

[16] Z. Baker and V. Prasanna, "Efficient hardware data mining with the Apriori algorithm on FPGAs," in *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2005. FCCM '05*, April 2005, pp. 3 – 12.

[17] ——, "An Architecture for Efficient Hardware Data Mining using Reconfigurable Computing Systems," in *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006. FCCM '06*, April 2006, pp. 67 –75.

[18] Y.-H. Wen, J.-W. Huang, and M.-S. Chen, "Hardware-Enhanced Association Rule Mining with Hashing and Pipelining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 784 –795, June 2008.

 ISBN: 978-1-61208-217-2