RC23970 (W0606-029) June 6, 2006
Mathematics

IBM Research Report

Using Secure Coprocessors for Privacy Preserving Collaborative Data Mining and Analysis

Bishwaranjan Bhattacharjee, Naoki Abe, Kenneth Goldman, Bianca Zadrozny[1], Vamsavardhana R. Chillakuru, Marysabel del Carpio[2], Chid Apte
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

[1]Universidade Federal Fluminense

[2]IBM Software Group

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. Ithas been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distributionoutside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g, payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

Using secure coprocessors for privacy preserving collaborative data mining and analysis

Bishwaranjan Bhattacharjee,
Naoki Abe,
Kenneth Goldman
IBM T.J.Watson Research Center
{bhatta,nabe,kgoldman}
@us.ibm.com

Bianca Zadrozny,
Universidade Federal Fluminense
bianca@ic.uff.br

Chid Apte,
IBM T.J.Watson Research Center
apte@us.ibm.com

Vamsavardhana R. Chillakuru,
Marysabel del Carpio
IBM
{vamsic,marysabel}
@us.ibm.com

ABSTRACT
Secure coprocessors have traditionally been used as a keystone of a security subsystem, eliminating the need to protect the rest of the subsystem with physical security measures. With technological advances and hardware miniaturization they have become increasingly powerful. This opens up the possibility of using them for non traditional use. This paper describes a solution for privacy preserving data sharing and mining using cryptographically secure but resource limited coprocessors. It uses memory light data mining methodologies along with a light weight database engine with federation capability, running on a coprocessor. The data to be shared resides with the enterprises that want to collaborate. This system will allow multiple enterprises, which are generally not allowed to share data, to do so solely for the purpose of detecting particular types of anomalies and for generating alerts. We also present results from experiments which demonstrate the value of such collaborations.

Categories and Subject Descriptors
H.2.8 [**Database Applications**]: *Data mining*

General Terms
Algorithms, Design, Security, Legal Aspects

Keywords
Privacy, Data Mining, Federation, Collaboration.

1. INTRODUCTION
An issue that has gained importance recently is the ability to cope with the contradictions between security and privacy on one hand and the requirement to share data across multiple enterprises on the other. This kind of collaboration helps in the detection of relevant trends and anomalies in the data. The requirement to collaborate is often mandated by legislations. For example in the USA, the Patriot Act [1] requires banks to analyze customer transaction data for anti-money laundering and other purposes. On the other hand, the Graham-Leach-Bailey Act [2] prohibits, in many cases, the sharing of data for any other purposes. This requirement could even extend to interactions between two lines of business (LOBs) in a company in some cases. Similarly in the field of health care, doctors and insurance companies need to protect the privacy of patient data, while the health care community as a whole can gain if that data can be pooled and analyzed for common good.

In this paper, we present a solution for privacy preserving data sharing and mining for such application areas. This solution allows the data to reside with the enterprises themselves and be processed in a federated setup in secure but resource constrained coprocessors like the IBM PCIXCC[3]. The transmission of data to the coprocessors is encrypted and is thus protected from eavesdroppers. Also, since the coprocessor is secure, the data which is decrypted and processed inside is protected from those sharing the data and any third party administering the solution. Further the data can also be joined, mined and analyzed in its original plain text. To facilitate mining in resource constrained coprocessors, we have developed and embedded memory light data mining methodologies. These provide the needed ability to detect trends and anomalies in shared data sets. They are run on data managed by a light weight database engine (like IBM Cloudscape[4]) with secure federation extensions.

A possible application area is Anti Money Laundering (AML) in an inter bank service center for due diligence. Money Laundering generally involves money transfer from one account to another, often spanning financial establishments. For this to be detected, these financial establishments would have to collaborate and try to determine suspicious patterns in the transactions. However, they are also concerned about the privacy of their data. A solution for sharing and mining of data while maintaining privacy would help in these cases.

Another application area is credit rating in an intra bank service center with multiple LOBs. Here we have two contradicting imperatives, namely, analyzing the data from multiple LOBs to know customers better and to keep in mind that there are legislations which limit LOBs from sharing data depending on the line of business.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Proceedings of the Second International Workshop on Data Management on New Hardware (DaMoN 2006), June 25, 2006, Chicago, Illinois, USA.
Copyright 2006 ACM 1-59593-466-9/06/25…$5.00.

The current state of the art in this field involves joining and mining over encrypted data on conventional machines. Examples include [5],[6],[7]. Queries over encrypted data have limitations for inequality joins, range queries and other operations. Methods to overcome them (like order preserving encryption) don't give the level of protection needed [8]. Encryption also causes problems for subsequent mining, since it generally does not preserve classes of distributions over numerical fields. Further there is no clear mechanism for fine grained access control of data thus generated, although some like [5], have role based access control for whole data sets.

In contrast, in this solution, the data is processed in plain text in the secure coprocessors as well as in the data sources, and thus does not have the limitations of the above scheme. Further, an enterprise is given access to the results relevant to it. For example in the case of banks sharing data for AML, a bank will come to know of only those transactions deemed suspicious to which that bank was a party and not all transactions in that category.

It should be noted that there is a crucial difference between our approach and processing over plain text in a conventional machine in a secure facility. The latter is used in [9]. This scheme is feasible only when one entity is making its data available for analysis. Here the entity modifies the results of queries to protect privacy. However when two or more independent parties share data, one has to trust the party administering the solution. This trust often does not go well with the privacy and secrecy rules that govern how sensitive information needs to be handled and thus this scheme would not work in that scenario. Our solution does not need the party administering the solution to be trusted and would work in a hostile environment.

Another approach to be contrasted with ours is the approach to privacy protection based on data perturbation. While this is an intriguing idea that introduced the topic of privacy in data mining, this approach comes with an intrinsic trade-off between predictive accuracy and privacy protection [10]. Also if there is a need to join data from multiple sources, data perturbation on the join columns would make it very problematic.

Another relevant work is [11], which dealt with secure sharing of data from multiple sources with access control on the IBM 4758 secure coprocessor. However in this case the data was in the form of flat files and all application logic was hand coded including any joins that needed to be done.

The rest of the paper is organized into the following sections. In section 2 we describe the IBM PCIXCC processor and its current and possible future usage. In section 3 we describe the architecture of the proposed solution. In section 4 we describe the light weight data mining methodologies which were developed. In section 5 we describe the database issues that arose and needed to be addressed. In section 6 we present experimental results which demonstrate the value of the work and finally in section 7 we provide concluding remarks and discuss future directions.

2. THE IBM PCIXCC PROCESSOR

The PCIXCC hardware is implemented in the form of a PCI-X adaptor card, with a secure module containing all security-related components. Figure 1 shows a photograph of the PCIXCC and Figure 2 depicts a block diagram of the card that includes the components in the secure module and those attached to the motherboard. The module is designed to meet the stringent security requirements of the FIPS 140-2 standard at its strongest level - level 4[12]. The internal components include a processor subsystem consisting of an IBM PowerPC 405GPr processor operating at 266 MHz, with 64 MB of dynamic random-access memory (DRAM) and 16 MB of flash-erasable programmable read-only memory (flash EPROM) for persistent data storage. Integrated peripheral devices on the processor chip include both Ethernet and serial port interfaces. There is also a high speed hardware cryptographic accelerator. Multiple PCIXCC cards can be used in a parallel system to tackle a given problem together.

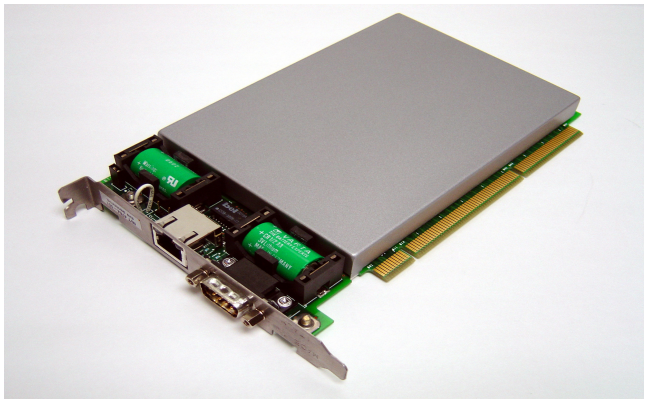


Figure 1: The PCIXCC

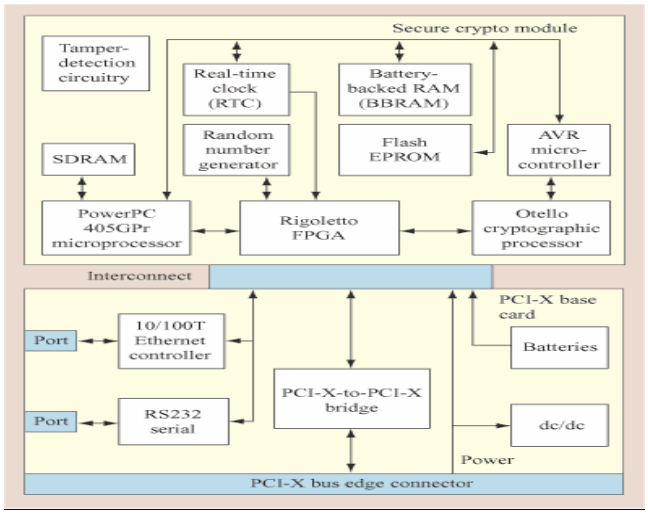


Figure 2: PCIXCC block diagram

The hardware cryptographic accelerator chip in the PCIXCC provides fast hardware implementation of the essential cryptographic algorithms used by the card. It supports the DES, and TDES symmetric encryption algorithms and the SHA-1 and MD5 secure hashing algorithms. In addition, the DES implementation includes both single-DES and TDES MAC support, conforming to ANSI X9.9 and ANSI X9.19. It also incorporates a public-key unit with modular math functions that are used to provide algorithms such as RSA.

The PowerPC 405GPr [25] used in the PCIXCC, is a low power consuming, 32-bit RISC processor targeted at and quite popular in embedded applications. Figure 3 shows a block diagram of the

processor. It incorporates a PowerPC 405 processor core, 16KB of separate Instruction and Data caches, 4KB on-chip memory, a PCI interface, an SDRAM controller, a 64-bit on-chip CoreConnect bus, a Fast Ethernet controller and other on-chip peripheral support. The 405 CPU operates on instructions in a five stage pipeline consisting of a fetch, decode, execute, writeback and load write-back stage. Its low power consumption of 0.72 W at 266 MHz translates to low heat produced. This is an important design consideration since the PCIXCC is completely encapsulated in a temper responding cover as can be seen in Figure 1. And so heat dissipation is an issue.

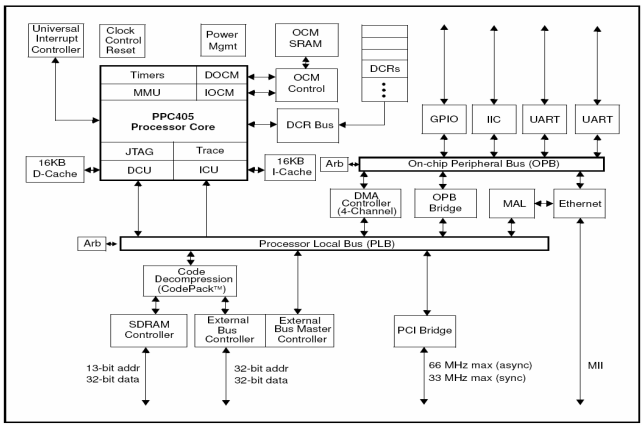


Figure 3: PowerPC 405GPr functional block diagram

The software stack on the PCIXCC is implemented as a layered design with a bootstrap loader at the lowest level and an application program at the highest level in the hierarchy. The application program is loaded into the flash EPROM after a security audit. The card uses an embedded Linux operating system that provides a subset of the features normally found in desktop or server Linux systems including shared library support, C and C++ run-time support, thread support and software floating-point support. More details about the PCIXCC can be found in [3].

Traditionally these types of secure coprocessors have been used as the keystone of a security sub-system, eliminating the need to protect the rest of the sub-system with physical security measures. For example, the PCIXCC is used in the IBM z990 mainframe for secure key cryptography. However these types of secure processors are becoming more and more powerful. Figure 4 shows the hardware trends for IBM secure processors over the years. Compared to its predecessor, the 4758 [13], the PCIXCC has 10 times more main memory, 2.5 times the processor clock speed, approx 9 times the Dhrystone 2.1 MIPS rating and better external connectivity via Ethernet. Further it runs a standard operating system like Linux in contrast to the CP/Q that the 4758 runs.

It is to be noted that the PCIXCC is still underpowered compared to current database servers. Floating point support is in software emulation only. Despite these, the PCIXCC certainly has characteristics which compare with a server node of yesteryears like the thin2 node in an IBM SP2 [14]. The thin2 node in a SP2, has been used in the past for TPCH benchmarks and commercial database workloads in conjunction with DB2 Parallel Edition [15]. It ran a 66 MHz POWER2 with an approximate Dhrystone 2.1 MIPS of 124, supported the AIX OS, anything from 64MB to 512MB of main memory and connection using Ethernet or optical switch. In comparison, the PCIXCC runs on 266 MHz with an approximate Dhrystone 2.1 MIPS of 404, supports 80MB of main memory, Ethernet connections and a Linux OS.

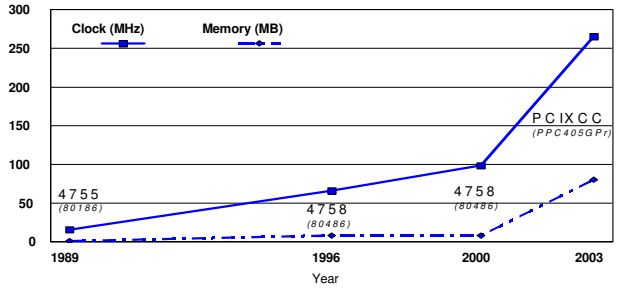


Figure 4: IBM Secure Processor hardware trends

These factors open up the possibility of using the new secure processors as semi independent processors for other applications. Such applications would require the security this hardware provides, would generate insights not available otherwise and could live with query performance power which might not be state of the art but is within the operating range for creating useful applications. This is the area of exploration of this project and solution.

3. ARCHITECTURE OF THE SOLUTION

Figure 5 and 6 give an outline of the solution. In our solution, the secure coprocessors are the main processing units rather than being slaved to a big server. They use the server to which they are connected as a file server. One could have multiple secure processors working in parallel to solve a given problem with one or more of them acting as coordinators. The enterprises sharing data would retain the data with them in relational tables, but are expected to allow their transmission to the database server in the secure processors in encrypted form. This is done using the industry standard Secure Socket Layer (SSL) protocol [26]. At the secure processor, this data will be decrypted, joined and processed with data from other enterprises. The overall architecture is basically a federated architecture with the secure processor being the federator and the enterprises acting as the data sources. In our current prototype we use one coprocessor but we believe that in principle the architecture can be extended to multiple secure processors working in parallel.

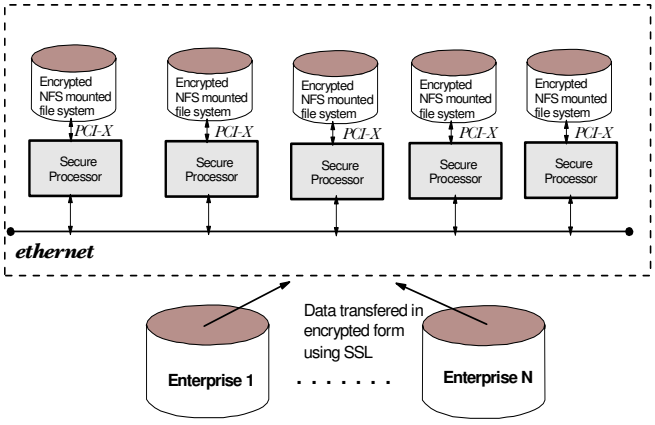


Figure 5: Proposed Solution Architecture

The SSL protocol provides security with a private and reliable connection. Symmetric cryptographic algorithms (e.g., DES [27]) are used for data encryption. The exact algorithm is decided during the process of an initial handshake. Message transport includes a message identity check using a keyed MAC. This is computed using secure hash functions (e.g., SHA [28], MD5 [29] etc). Most of the algorithms mentioned are supported in the special cryptography accelerator in the secure processor. This will help to significantly speed up the execution of the SSL protocol.

In our prototype, each table that is being shared by the enterprises is represented by a Virtual Table Interface (VTI) at the secure processor. Queries on the secure processor are run on these VTIs. The VTIs in turn make Secure JDBC calls to connect to the enterprises, get the data in encrypted form, decrypt it and pass it on to the query processing functionality of the Cloudscape database server. More details can be found in section 5.

Once the data is available, it has to be mined to determine patterns. Generic data mining algorithms tend to be memory intensive and using them would be difficult on the resource constrained processor. A key challenge we have undertaken in this project is the development of light weight data mining algorithms. They take much less memory to run and are very easily parallelizable. More details can be found in section 4.

For access control of the mined data and its results, we plan to use a version of the Matchbox access control mechanism described in [11]. Here, information providers, users and third parties use digitally co-signed contracts that enforce the sharing restrictions. These access control mechanisms would be over and above what is available at a database level.

Also the database engine as well as the mining application might generate temporary data which will have to be stored locally on the file system. This data needs to be secured. To this end we propose to enhance the file system to an encrypted file system. To speed up encryption/decryption of contents, it will tap the special cryptographic chip in the secure processor. These file systems can be NFS mounted into the secure processor from a file server connected via the PCI bus. Note that any index created on local tables on this file system would be created on plaintext and its pages encrypted before storing. It would not be an index on encrypted data.

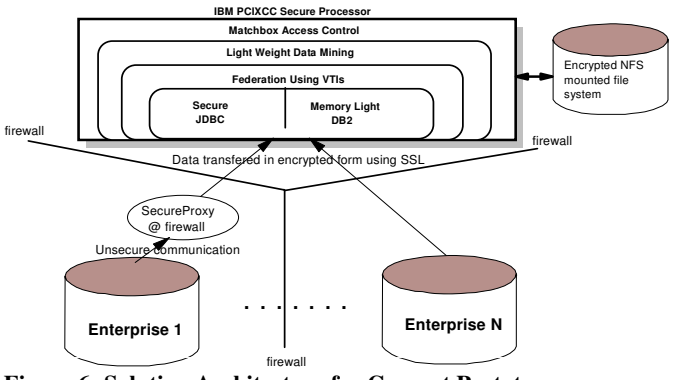


Figure 6: Solution Architecture for Current Prototype

4. MEMORY LIGHT DATA MINING

The memory light data mining [16] we embedded in our secure federated mining system consist of two functionalities: cost-sensitive classification and outlier detection. Here, we describe the cost-sensitive classification algorithm we employed, as well as empirical evaluation of the method's performance on a benchmark data set.

4.1 Cost-sensitive classification

Cost-sensitive learning refers to the problem of learning classifiers in which non-uniform costs are associated with different types of misclassifications. For many data mining problems of practical importance, it is essential to take this aspect into account. Examples of such applications include credit rating, fraud detection, churn analysis, and targeted marketing. For example, frauds are so rare that straightforward application of classification for fraud detection can result in a classifier that always predicts the most common class (non-fraud), but when they occur, frauds are a magnitude more costly and thus it is important to classify them correctly.

A formulation of the cost sensitive learning problem is by the use of a cost matrix C(i,j,x). This specifies how much cost is incurred when misclassifying a label j as i depending on an instance (or example) x [17]. The goal of a cost-sensitive learning method is to minimize the expected cost. For binary labels, C(i,j,x), can be reduced to a simpler formulation in terms of an importance number per example [18]. This is possible by associating a single number indicating the importance of an example (x,j), given by C(1,j,x) - C(0,j,x). This conversion allows us to formulate the problem, in which we assume that examples are drawn independently from a distribution D with domain $X \times Y \times C$ where X is the input space to a classifier, Y is a (binary) output space and C is the set of possible importance weights (extra cost) associated with mislabeling that example. The goal is to learn a classifier $h: X \rightarrow Y$ which minimizes the expected cost, $E_{x,y,c \approx D}[c \cdot I(h(x) \neq y)]$ given training data of the form: (x,y,c) , where $I(\cdot)$ is the indicator function that has value 1 in case its argument is true and 0 otherwise.

A basic folk theorem, which was proven in [17] states that if we have examples drawn from the distribution modified with multiplicative factors proportional to cost: $D'(x,y,c) \approx c \cdot D(x,y,c)$ then optimal error rate classifiers for D' are optimal cost minimizers for data drawn from D .

Proposition

For all distributions, D, there exists a constant $N = E_{x,y,c \sim D}[c]$ such that for all classifiers, h: $E_{x,y,c \sim D'}[I(h(x) \neq y)] = (1/N) E_{x,y,c \sim D}[c \cdot I(h(x) \neq y)]$, where we used D' to denote $D'(x,y,c) \equiv (c/(E_{x,y,c \sim D}[c])) \cdot D(x,y,c)$.

4.2 A Memory-Light Method of Cost-sensitive Learning

The ``folk" theorem mentioned above suggests an obvious method of converting from one distribution of examples to another to obtain a cost-sensitive learner by re-weighting the example distribution. Since straightforward sampling methods such as resampling with replacement do not work well in this case, we

make use of a sampling scheme called rejection sampling [19]. This allows us to draw samples independently from the distribution D', given samples drawn independently from D. In rejection sampling, samples from D' are drawn by first drawing samples from D, and then keeping the sample with probability proportional to D'/D . Here, we have $D'/D \approx c$, so we accept an example with probability c/Z , where Z is some constant chosen so that $\max_{(x,y,c)\in S} c \leq Z$. Note that rejection sampling results in a set S' which is generally smaller than S .

Using *cost-proportionate rejection sampling,* just introduced, to create a set S' and then using a learning algorithm A(S') is guaranteed to produce an approximately cost-minimizing classifier, as long as the learning algorithm A achieves approximate minimization of classification error.

From the same original training sample, different runs of cost-proportionate rejection sampling will produce different training samples. Furthermore, the fact that rejection sampling produces very small samples means that the computational time required for learning a classifier is generally much smaller.

We take advantage of these properties to devise an ensemble learning algorithm called "costing" based on repeatedly performing rejection sampling from S to produce multiple sample sets $S_{1'},...,S_{m'}$, and then learning a classifier for each set. The output classifier is the average over all learned classifiers.

Costing(Learner A , Sample Set S , count t , normalization constant Z)

1. For i=1 to t do

 1.1. S'= rejection sample from S with probability c/Z.

 1.2. Let $h_i \equiv A(S')$

2. Output $h(x)= \text{sign}(\Sigma_{i=1}^{t} h_i (x))$

Figure 7: The ``Costing'' algorithm.

Note that despite the extra computational cost of averaging, the overall computational time of costing is generally much smaller than for a learning algorithm using sample set S (with or without weights). In particular, this is the case if the component learning algorithm being employed has a running time that is superlinear in the number of examples, which is generally the case.

Also a major advantage with this scheme is that the iterations could run in parallel on a set of processing units. Each unit could independently learn a subset of classifiers from t and then these classifiers could be averaged at one processing unit.

4.3 Empirical evaluation using a benchmark

We show empirical results using a real-world dataset from the direct marketing domain used in the KDD-98 competition [20]. It contains data on people who have made donations in the past to a particular charity, and the decision-making task is to choose which donors to mail solicitations for donation, with the goal of maximizing the total profit obtained in the mailing campaign. The dataset is divided in a fixed way into a training set and a test set, each of size approximately 96,000.

Figure 8 shows the results of applying costing on the KDD-98 dataset, using Quinlan's C4.5 decision tree algorithm [21] as the base learner. The graph plots the profits achieved by the obtained mailing rules, as a function of the number of iterations. In this experiment, each resampled set has only about 600 examples, because the importance of the examples varies from 0.68 to 199.32 and there are few ``important'' examples. With t=200, our method yields profits around 15,000 dollars, which is exceptional performance for this dataset. (For example, the KDD cup 98 competition achieved 14,712 dollars [22].)

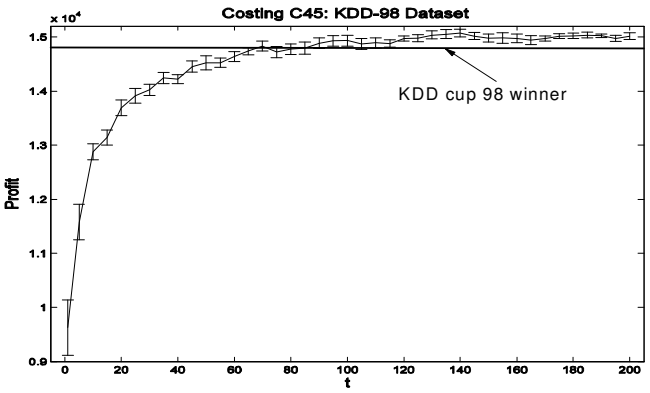


Figure 8: Profits obtained by Costing applied on C4.5, as a function of the number of sampling iterations.

These results are particularly remarkable from the point of view of reduction in memory requirement: only hundreds of examples need to be processed in iteration, as compared to the original data size of approximately 100 thousand. Even with this drastic reduction in memory requirement, costing manages to achieve state-of-the-art predictive performance.

5. DATABASE CHALLENGES

The current prototype is centered on Cloudscape as the database engine [4] running in the secure processor. Cloudscape is IBM's Java based small footprint database server and is used as a data federator in this case. To get Cloudscape working on the PCIXCC, the software stack was first enhanced to support a Java Virtual Machine called J9 on which Cloudscape could run.

Cloudscape provides a construct for federation called Virtual Table Interface (VTI). These are used to access external data sources. Internally the VTIs make Java Data Base Connectivity (JDBC) calls to connect and get data from these data sources. For secure federation, additional functionality had to be built into the VTIs to handle SSL connections with the data sources. This functionality is known as Secure JDBC. It was developed as part of the project on the Java Common Client (JCC) driver of the DB2 family. This was done using the IBM Java Secure Socket Extension (JSSE) library on top of the Java Runtime Environment (JRE).

Some of the database server engines which might be used at the data stores do not support secure JDBC or secure Client/Server communications. An example is DB2 LUW V8. To tackle this, we developed a proxy server which could be used between the secure processor and the actual data server. This proxy could sit on the firewall of the enterprise and would be able to manage communication between the secure processor and the data server. Figure 6 shows a possible configuration with these proxy servers.

Once these infrastructure issues were taken care of, the key challenge from the database point of view is how to efficiently run

the database operations while maintaining security and privacy. This is an instance of data federation where the data sources cannot talk to each other directly and the federator is resource constrained. Since the data sources controlled by the enterprises would be in plaintext, all conventional database optimization techniques would be applicable there including indexes, materialized views etc.

From the federator, as much of the computation and predicates as possible should be pushed down to the data sources. Equality joins could be best handled by a variation of Sort Merge join. In Sort Merge join the data would come in sorted on the join columns to the federator from the data sources. The resource constrained federator would just need to join the sorted streams. Inequality joins can be handled by a Nested Loop join variation where one extracts tuples from the outer and uses the join column values to generate queries for the inner data source. For both join types one needs to keep privacy and security issues in mind while implementing the join algorithms.

For the memory light data mining application, two database issues are important. The first is sampling. This could be pushed down to the data sources. The most relevant sampling technique is Key Value Sampling. This is not supported on some commercial database engines. In place of that one could materialize the join columns in the coprocessor and generate queries driven by samples from it.

Another issue is exploiting multiple secure processors. One could divide the data mining iterations among the processors for parallelism as explained in Sec. 5. But in general, a query could be processed by multiple processors by assuming a logical hash partition of a table residing on the data sources is going to be processed by a processor. For example, with n secure processors, one could assume records with join column jn, will be processed at node k if k = (jn%n). This could be implemented by additional predicates on the VTIs. The resulting mini joins could then be fused and processed at a coordinator secure coprocessor.

6. EXPERIMENTAL EVALUATION: CREDIT RATING IN A FEDERATED SETUP

To demonstrate the effectiveness of our solution, we ran an experiment on our prototype using the credit rating data set from the PKDD-99 Discovery Challenge [23]. It has relational tables that contain customer data on banking transactions and credit card transactions, as well as loan default information. We set up our experiments so that the loan data is stored in a database on one machine, whereas the banking and credit card data resides on another machine connected through a network as shown in Figure 9. This setup corresponds to the real world scenario in a company with two lines of businesses, namely banking and loan departments, own their respective customer data in separate computing environments. Regulations may restrict them from freely sharing their data, except for the purpose of due diligence, namely in making sure that the banks know the customers well enough to avoid awarding loans to high risk customers.

To this end, using secure federation, we first join data from multiple databases for various accounts of an individual. Then we convert the information to a feature vector. An example feature vector would be <*sum of loans outstanding, sum of current account balances, current income*>. On this, we apply memory light data mining to generate the credit risk rating rules.

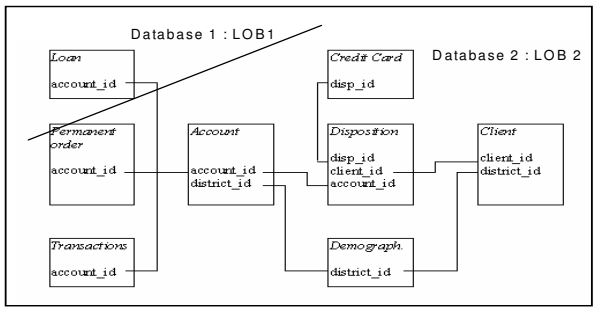
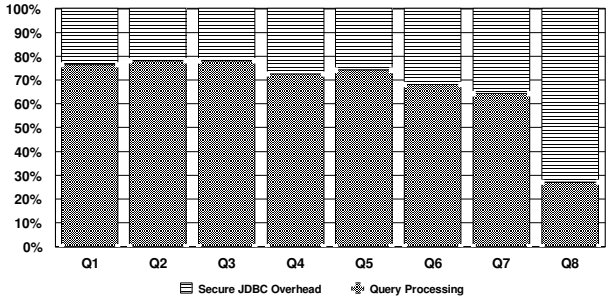


Figure 9: The PKDD-99 Discover Challenge data set in a federated set up.

The data mining code issued 8 different types of queries on the VTIs. Seven of the query types had two or multi way equality joins on data owned separately by the departments. The breakup of time spent in the queries is shown in Figure 10. For the large, multi table join queries (Q1, Q2, Q3), the overhead of encryption/decryption during SSL was ~ 21% and for the single table query (Q8) it was about ~70%. This is minor compared to the overheads of the scheme using commutative encryption mentioned in [7]. Using the hardware cryptographic accelerator in the PCIXCC would bring down this overhead significantly. Also, all the queries could be very easily parallelized to run on multiple secure coprocessors using the schemes described in section 6.



.

Figure 10: Breakup of time in the various query sets

The results of the experiment as indicated in Figure 11 are quite intriguing. As a baseline we note that awarding a loan to every customer results in loss of about CZK 2,000,000. If we use a state-of-the-art, cost-insensitive classification method of applying bagging over C4.5 [21, 24] to obtain rules for awarding loans, it results in an improved loss of about CZK 500,000. In contrast when we used our memory-light data mining method (costing) implemented in a secure federated environment, its rules achieved a profit of CZK 600,000. This is remarkable, especially since the comparison method, i.e. bagging over C4.5, is not memory-light, requiring over a hundred times more sample size in an iteration.

Decision Making Method	**Profit/Loss (In CZK)**
Loan to everybody	Loss of ~ 2,000,000
Bagging over C4.5	Loss of ~ 500,000
Using Our Solution	**Profit** of ~ 500,000

Figure 11: Performance comparison for the experiment

The predictive performance of our method is due to our ability to join the separate data from the loan and the banking departments. This is visible in one of the output decision trees shown in figure 12. This decision tree is obtained by combining the accounts and credit card information (e.g. features like min_balance6 and credit card) from the banking and credit card departments, and the past loan default information from the loan department.

```
joint-account = Yes:  GIVE LOAN
joint-account = No:
|  credit_card = Classic : GIVE LOAN
|  credit_card = Junior  :  GIVE LOAN
|  credit_card = Gold    : GIVE LOAN
|  credit_card = No:
|  |  min_balance6 <= 7186.4 : DON'T GIVE LOAN
|  |  min_balance6 > 7186.4 :
|  |  |  avg_balance1 <= 42188.6 : GIVE LOAN
|  |  |  avg_balance1 > 42188.6 : DON'T GIVE LOAN
```

Figure 12: A sample decision tree reached in the experiment

7. CONCLUSION

Secure processors are becoming increasingly powerful with better hardware miniaturization. This opens up the possibility to use them for non traditional uses. In this paper we have proposed a privacy preserving data mining and sharing architecture based on secure processors which will allow multiple entities to collaborate and gain insights from their shared data. We have discussed some of the key challenges and ways to tackle them. We have also demonstrated the usefulness of the architecture with an example. Future enhancements include building the framework to tap multiple secure processors; memory light, privacy preserving, join methods for inequality joins; pushing down sampling to the data sources by queries and validation in a customer scenario.

8. REFERENCES

[1] Patriot Act, http://thomas.loc.gov/cgi-bin/bdquery/z?d107:h.r.03162.

[2] Graham-Leach-Bailey Act,http://www.ftc.gov/privacy/glbact.

[3] T.W. Arnold, L.P. Van Doorn,"The IBM PCIXCC : A new cryptographic coprocessor for the IBM eServer", IBM Journal of Research and Development, Vol 48, May 2004

[4] Cloudscape, http://www-306.ibm.com/software/data/cloudscape.

[5] D.P. Hansen, C. Daly, K. Harrap, J. Jacquet, M. O'Dwyer, C. Pang, J. Ryan-Brown, "Health Data Integration (HDI): Research Software to Commercial Product", Australian Software Engineering Conference , 2005

[6] Entity Analytics Solutions, http://www.ibm.com/software/data/db2/eas

[7] R. Agrawal, D. Asonov, R. Srikant, "Enabling Sovereign Information Sharing Using Web Services", Proceedings of the SIGMOD 2004

[8] M. Kantarcioglu, C. Clifton, "Security issues in querying encrypted data", Technical Report TR-04-013,Purdue University, 2004

[9] Privacy Preserving Analytics, CSIRO Annual Report 2004-5 and CSIRO "PPA for Health Data" brochure

[10] C. Clifton, W. Du, M. Atallah, "ITR: Distributed Data Mining to Protect Information Privacy", Purdue University

[11] K. Goldman, E. Valdez: "Matchbox: Secure Data Sharing", IEEE Internet Computing, 8(6) 2004, pp 18-24.

[12] FIPS Standards, http://csrc.nist.gov/cryptval/140-2.htm

[13] IBM 4758, http://www-.ibm.com/security/cryptocards/pcicc/overview.shtml

[14] T. Agarwala, J.L.Martin, J.H.Mirza, D.C.Sadler, D.M.Dias, M. Snir,"SP2 system architecture", IBM System Journal, Volume 34, No. 2, 1995

[15] C.K. Baru, G. Fecteau, A. Goyal, H. Hsiao, A. Jhingran, S. Padmanabhan, G.P. Copeland, W.G. Wilson,"DB2 Parallel Edition", IBM Systems Journal, Vol. 34, No. 2, 1995

[16] N. Abe, C. Apte, B. Bhattacharjee, K. Goldman, J. Langford, B. Zadrozny, "Sampling Approach to Resource Light Data Mining", SIAM Workshop on Data Mining in Resource Constrained Environments 2004

[17] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown", Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, pp 204-213, 2001.

[18] B. Zadrozny, J. Langford and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting", Proceedings of the Third IEEE International Conference on Data Mining, pp 435-442, 2003

[19] J. von Neumann, "Various techniques used in connection with random digits", Applied Mathematics Series, 12, pp 36-38, National Bureau of Standards, 1951.

[20] S.S. Bay,"UCI KDD Archive", Department of Information and Computer Sciences, University of California, Irvine, http://kdd.ics.uci.edu/, 2000.

[21] J. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, CA, 1993

[22] KDD-cup-98 Results, http://www.kdnuggets.com/meetings/kdd98/kdd-cup-98-results.html.

[23] PKDD'99 Discovery Challenge: A collaborative effort in knowledge discovery from databases, http://lisp.vse.cz/pkdd99/chall.htm.

[24] L. Breiman, Bagging Predictors, Machine Learning, 24, pp123-140, 1996

[25] PPC 405GPr Embedded Processor Data Sheet, AMCC, 2005

[26] A. Freier, P. Karlton, P. Kocher, "The SSL Protocol, Version 3.0",Transport layer Security Working Group, 1996, http://wp.netscape.com/eng/ssl3

[27] ANSI X3.106, "American National Standard for Information Systems-Data Link Encryption", American National Standards Institute, 1983

[28] R. Rivest, "The MD5 Message Digest Algorithm", April 1992

[29] FIPS Standards, http://www.itl.nist.gov/fipspubs/fip180-1.htm