

BronzeGate: Real-time Transactional Data Obfuscation for GoldenGate

Shenoda Guirguis
Computer Sc. Dept.,
University of Pittsburgh
shenoda@cs.pitt.edu

Alok Pareek
Vice President, Product Management
Oracle Product Development
alok.pareek@oracle.com

ABSTRACT

Data privacy laws have appeared recently, such as the HIPAA laws for protecting medical records, and the PCI guidelines for protecting Credit Card information. Data privacy can be defined as maintaining the privacy of Personal Identifiable Information (PII) from unauthorized accessing. PII includes any piece of data that can be used alone, or in conjunction with additional information, to uniquely identify an individual. Examples of such information include national identification numbers, credit card numbers, as well as financial and medical records. Access control methods and data encryption provide a level of data protection from unauthorized access, however, it is not enough; it does not prohibit identity thefts. It was reported that 70% of the data privacy breaches are internal breaches that involve an employee from the enterprise who has access to some training or testing database replica, which contains all the PII. In addition to access control, we need techniques to obfuscate (i.e., mask or dim) the datasets used for training, testing and analysis purposes. A good data obfuscation technique would, among other features, preserve the data usability while protecting its privacy. This challenge is further complicated when real time requirements are added. In this paper we present BronzeGate: Obfuscated GoldenGate, the GoldenGate's real-time solution for transactional data privacy while maintaining data usability. BronzeGate utilizes different obfuscation functions for different data types to securely obfuscate the data, on real-time, while maintaining its statistical characteristics.

General Terms

Algorithms, Design, Management, Security, Human Factors, Legal Aspects.

Keywords

Data Obfuscation, Masking, Privacy, Security, Usability, Real-Time Transactional Data Management.

1. INTRODUCTION

Data Privacy is no more an optional feature; it is a requirement by any data management system to preserve the privacy of the data of the users of the system. Recently, privacy laws have appeared, such as the HIPAA laws [2] for protecting medical records, and

the PCI guidelines [3] for protecting Credit Card information. Data privacy, AKA information privacy, can be defined as maintaining the privacy of personal identifiable information or data from unauthorized accessing. Data privacy refers to developing relationship and interaction between technology and the privacy of personally identifiable information (PII) that is collected, stored, and shared by organizations. PII includes any piece of data that can be used alone, or in conjunction with additional information, to uniquely identify an individual. Examples of such information include first and last names, social security numbers, national identification numbers, addresses, date of birth, phone numbers, email addresses, driver's license numbers, credit card numbers, financial and medical records, etc.

Data Security has been preserved through access control. Although access control methods provide a level of data protection, it is not enough. Access control methods, in addition to data encryption, protect data from unauthorized access. However, it does not prohibit identity thefts. It was reported that 70% of the data privacy breaches are internal breaches that involve an employee from the enterprise who has access to some training or testing database replica, which contains all the PII [1].

Therefore, there is a strong need for techniques that would prevent such identity thefts. Ideally, we need a technique that would protect the PII from unauthorized access, and allow an access for analysis, testing and training purposes, while maintaining its usability. The challenge here is the contradicting requirement of a usable dimmed copy of the data that, yet, does not breach the privacy of the data. At GoldenGate Co, we currently have couple of use-cases of our clients that have these requirements, including a large financial credit card enterprise.

Data Obfuscation (DO) is a broad term that refers to any data manipulation technique used to induce ambiguity to the data, desensitize it to be of no sense, yet usable, and thus preserving its privacy.

1.1 Desired Properties of a Data Obfuscation Technique

The two main requirements of a DO technique are Data Privacy and Usability. Data Privacy refers to the fact that the PII are secured and concealed upon applying the DO technique to the data. Usability refers to the fact that the transformed data is still useful and maintains the main statistical and semantic properties of the original data. In addition, there are a set of desired properties:

1. "Providing access to the confidential attributes should provide the intruder with no additional information" [4]. In other words, the ability to predict the original data given access to the obfuscated data should be the same as it is without access to the obfuscated data.

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, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. EDBT 2010, March 22-26, 2010, Lausanne, Switzerland. Copyright 2010 ACM 978-1-60558-945-9/10/0003 ...\$10.00.

2. The DO technique should be irreversible. It should never be possible to use it to retrieve the original sensitive data given the technique and the obfuscated data.
3. Semantics and referential integrity must be maintained.
4. Obfuscation must be a repeatable process to guarantee consistency. This means that every time a data item is being obfuscated, it is obfuscated to the same obfuscated data item.

All these requirements make the task of obfuscating the data efficiently a real challenge. It is even more challenging when real-time requirements are added as in the motivating example below.

1.2 Motivating Example

Consider the case when GoldenGate [5] software is utilized to replicate bank transactional data across heterogeneous sites, where one copy of the data is replicated to a third party site to be used for real-time analysis purposes, say for fraud detection for instance. One way to do so is to replicate the data, then apply an existing obfuscation technique in an offline fashion and then use the obfuscated copy for analysis. Note that a mapping between original and obfuscated data items is needed in this example. This can be maintained securely encrypted at the original data host. This solution, although relatively simple, it does not satisfy the real-time requirements of the fraud detection. In addition, a copy of the original data is being copied and stored at a third party site before it is being obfuscated, which is a huge security threat. Thus, a strong need for a real-time transactional data obfuscation technique is needed: a technique that satisfies all desired properties of obfuscation techniques in addition to satisfying the real-time requirements.

In this paper, we present BronzeGate, which is the GoldenGate's real-time transactional data obfuscation solution. BronzeGate utilizes different obfuscation functions for different data types to securely obfuscate, on real-time, the data while maintaining statistical characteristics of the data, for testing and analysis purposes.

Road map: The rest of this paper is organized as follows. Section 2 furnishes the required background and Section 3 presents BronzeGate solution. Sketched analysis of BronzeGate is given in Section 4, while Section 5 provides sample experimental results. The paper is concluded in Section 6.

2. Background

Many techniques have been proposed for Data Privacy such as: 1) Data Randomization: which adds noise to the data, 2) Data Anonymization: which uses generalization and suppression to make the data ambiguous, 3) Data Swapping: which involves ranking data items and swapping records that are close to each other, 4) Geometric transformation: which uses transformations such as rotation, scaling, and translation for distorting the data, and 5) Nearest Neighbor Data Substitution: which uses Euclidean distance to define neighbors, and then perform swapping.

Some of these techniques apply to only certain data types. For example, the Geometric Transformation techniques apply only to numerical data. The majority of these techniques were developed for privacy protection for data mining and analysis, for which there are no real-time requirements. To the best of the authors knowledge, all these techniques involves an offline analysis phase, at which the statistical characteristics of the data

set is captured, and used to guide the obfuscation, in order to maintain these statistical characteristics.

In GoldenGate software, transactional data is being replicated on real-time fashion, and hence, a real-time obfuscation technique is needed. BronzeGate is a suite of techniques for obfuscating different data types. For numerical data, we propose a technique that is based on both Geometric Transformation, namely GT-NeNDS [1] and Anonymization. We explain these two techniques in more details next.

2.1 Numerical Data Obfuscation

GT-NeNDS [1] is the state of the art in numerical data obfuscation that is designed for clustering mining. We extend GT-NeNDS to make it applicable on Real-Time by applying Anonymization, which adds to the Data Privacy, and increases irreversibility, at the expense of data loss. However, this data loss is controlled as explained in Section 3.1.

2.1.1 Anonymization Approach:

Anonymization techniques map multiple data items into one. For example, it replaces the date with the month and year only. This generalization involves a loss of information, but data stays consistent. K-anonymity aims at mapping at maximum k data items into one representing data item. Anonymization techniques are irreversible, since there no way to know the original data item.

2.1.2 GT-NeNDS Approach:

GT-NeNDS stands for Geometric Transformation – Nearest Neighbor Data Substitution. GT techniques include scaling, rotating, and translation, these preserve data characteristics. NeNDS technique was proposed for privacy preservation for Clustering Mining applications. It proceeds like this: it clusters the original dataset into sets of neighbors. Neighborhood is determined using Euclidean Distance. Each data item in a neighbors' set is replaced by the nearest neighbor in this set, in a way such that no swapping occurs, using special data structures. Thus, statistical properties of the original data are preserved. NeNDS introduce a degree of obfuscation by replacing a data item with its nearest neighbor. GT-NeNDS aims at securing the data by further obfuscating the nearest neighbor using the GT techniques.

2.1.2.1 GT-NeNDS doesn't fit Real-Time setting!

GT-NeNDS does not fit in the real-time requirements due to the following reasons. First, to construct the sets of neighbors, the algorithm needs a pass through all the data, which is not feasible in real-time settings. Second, substituting a data item with its nearest neighbor means that the substitution is not repeatable because neighbors changes with insertions and deletions. To overcome these shortages, we propose GT-ANeNDS, and extension to GT-NeNDS.

3. BronzeGate solution

In this section, we introduce our proposed GT-ANeNDS technique, which overcomes GT-NeNDS' real-time limitations, and leverage the level of data privacy. We then discuss the different obfuscation techniques proposed for different data types, which together form our BronzeGate solution for real-time transactional data obfuscation.

```

GT-ANeNDS Algorithm: High Level
Input: dataset specifications (data-type,
  histogram, and semantics)
Input: Transactional data item (transaction-
  ID,value of data item)
Output: obfuscated value.
BEGIN
  Based on the semantics, determine the
  distance between the origin and the item
  value.
  Based on this distance, histogram and the
  semantics, pick the nearest neighbor.
  Based on semantics, apply the proper GT
  technique to the nearest neighbor.
  Return the obfuscated value.
END

```

Figure 1: GT-ANeNDS Algorithm: High Level

3.1 GT-ANeNDS:

GT-ANeNDS combines Anonymization and NeNDS techniques, which yields to gain: efficiency, real-time adherence, repeatable mapping, and higher level of data privacy. This comes at the expense information loss. However, this loss is controlled so that the data usability is not affected. GT-ANeNDS can be applied to any data type for which a distance function can be defined. We first give the higher level view of the algorithm then we explain it using numerical data type. In the discussion hereafter, by dataset we refer to a field, or a column, in the original database schema.

Figure 1 lists the main steps of the GT-ANeNDS approach. The Input to the algorithm consists of the new transactional data item, and the meta-data. The meta-data consists of: data-type, histogram and semantics. Below is the description of each.

Data-Type: The data-type is the regular database type, i.e., numerical, text, timestamp, etc. In addition to the semantics, datatype is used to determine the technique to use.

Histogram: We use the term histogram in a generic way to refer to the data structure that is incrementally maintained. A detailed discussion on histograms is presented soon.

Semantics: The semantics of each data set is a record of the following information whenever applicable:

- Data-Sub-Type: for numerical data, the sub-type defines whether the data are general, or identifiable. Where identifiable data can identify the person, such as the national ID number, SSN, etc.
- Euclidean distance Function: the function to be used to calculate the Euclidean distance between two values.
- The Origin point: the reference point of this data set.

Given the data-type and the semantics, the appropriate obfuscation technique is determined. In case it is GT-ANeNDS, the origin-point and the Euclidean distance function determine the appropriate bucket in the histogram, and the nearest neighbor therefore. Next, GT function is applied to the nearest neighbor, generating the obfuscated value. Next, we illustrate how the GT-ANeNDS works in case of numerical data types.

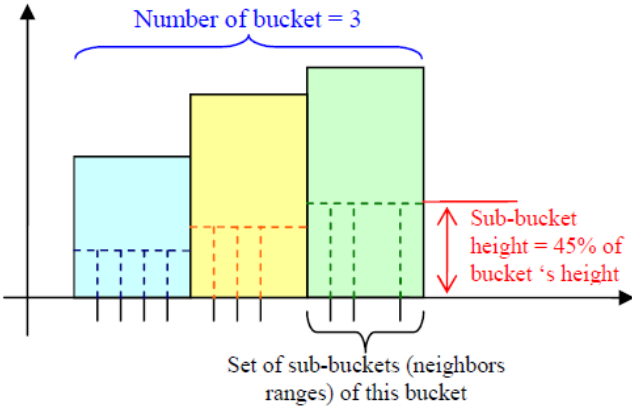


Figure 2: Histogram for a Numerical: general data type.

3.1.1 Numerical Data

For general numerical data (i.e., non ID's such as bank account balance), we use equi-width histograms that splits the range of the data items distances into regions of the same width (i.e., range) to define the set of neighbors. Each bucket's range is divided into a set of equi-height sub-buckets. The bucket's width and the subbucket's height are systems parameters set by the administrator. Histograms are built by scanning the current database shot once. The histogram decomposition is illustrated in Figure 2. As shown, the number of neighbors for each bucket depends on the height of the bucket and the position of these neighbors depends on the values distribution in this range.

Note that the horizontal axis is not the data value; however, it is the distance from the origin point. The vertical access is the frequency. This is introduced to be able to identify the nearest neighbor without the need to maintain any summary about the data values within each bucket.

The GT-ANeNDS proceeds as follows. First, the distance between the original data item's value and the origin point is calculated, determining where in the histogram this data item falls. Second, the nearest neighbor point in the histogram is determined. The neighbors set, is the set of points determining sub-buckets' ranges within the same bucket this point belongs to. Finally, geometric transformation is applied to the nearest neighbor, generating the obfuscated value.

The only difference to the GT-NeNDS is that GT-ANeNDS uses a fixed set of neighbors for each bucket, which yields to map more than one original data value to the same obfuscated value, i.e., Anonymization. By fine tuning the bucket widths and the sub-bucket heights, the statistical characteristics of the original data are minimally impacted, as the experimental section shows.

3.1.2 Boolean Data

For Boolean data-type, the same approach is used but we simply have two buckets only and no sub-buckets. Therefore, we maintain in this case two counters for each bucket. To obfuscate a value, the new value is randomly drawn with probability to have the same ratio of the two values. For example, if it is a Gender field and the counters are: ten females and seven males, then the obfuscated value is set to M (i.e., male) with probability 7/17.

```

Special Function 1: Obfuscating identifiable
numerical data
BEGIN
    temp1 ← Apply FaNDS to digits of input.
    temp1 ← Rotation(temp1)
    temp2 ← temp1 + original number.
    temp2 ← Truncate(temp2, length(input))
    For i=1 to length(input)
        Temp3[i] ← randomly pick between temp1[i]
        and temp2[i] based on input[i].
    Next i
    return Temp3 as the obfuscated data.
END

```

Figure 3: Special Function 1: Algorithm to Obfuscate Identifiable Numerical Data

3.1.3 Identifiable Numerical Data

When a numerical value is a key, such as national identification number, Anonymization is not valid, as it will result in distortion of the referential integrity constraints. We therefore propose Special Function 1, illustrated in Figure 3.

Opposed to NeNDS, we use FaNDS technique, which stands for Farthest Neighbor Data Substitution. It is exactly same as NeNDS except that we substitute the data item with its farthest neighbor. Each digit of the original value is treated as a separate value to obfuscate. The set of digits are used as the neighbors from which the farthest neighbor is chosen to replace the original digit. Next, rotation is applied for each replaced digit and saved in a temporarily variable. This rotated number that results from replacing each digit in the original key and then rotating it is being added to the original key value and result is truncated to the key length and saved in a second temporarily variable. Finally, the obfuscated key is generated by randomly picking each digit from the two temporarily variables.

3.1.4 Date Data

For date data type, neither GT-ANeNDS nor Special Function 1 fits, because of the semantics of the date. Therefore, we propose Special Function 2, to obfuscate date and timestamp data types. The function basically utilizes controlled randomness to obfuscate each component of the date, i.e., the day, month and year.

3.1.5 Other Data Types

In Table 1, we summarize the possible data-types, semantics, and which technique BronzeGate uses to obfuscate each data type. BronzeGate allows the user to overwrite these default selections and to define a user-defined obfuscation function. The metadata about which technique to be used and its parameters could be stored in the original database itself, or in a parameters file.

3.2 BronzeGate Architecture

BronzeGate architecture is displayed in Figure 5 which shows the regular GoldenGate’s simple replication architecture [5]. BronzeGate lies in the userExit process, which – in GoldenGate solution - performs user defined customized transformations to the replicated transactions. BronzeGate is hence a special type of userExit process, where the task is performs the required obfuscation on the fly.

Table 1: Input to Obfuscation Module

Data-Type	Semantics	Obfuscation Technique
Numerical	General	GT-ANeNDS GT technique: rotation
Numerical	Identifiable (numbers or text) Such as: SSN, Credit Card, driving license number, etc.	Special Function 1
Date	General	No Obfuscation OR Special Function 2.
Large Objects	General (Medical Reports, X-Rays, etc.)	No Obfuscation.
Boolean	General (s.a.: true/false, gender, etc.)	Randomly set, with certain probability.
Text	Identifiable (Such as: Names and contacts.)	Random replacement from a “Dictionary”.
Text	General	No Obfuscation
Any other data type		GT-ANeNDS if applicable.

As shown in Figure 5, the BronzeGate process runs at the original database site, to obfuscate the transactional data before they are shipped to the replicate site. BronzeGate process is activated by the Capture process, which monitors the original database. Whenever a transaction is committed to the original database, Capture process will capture this change and signals BronzeGate Process to handle this transaction. BronzeGate, in turn, uses the parameters file, histograms, and dictionaries to obfuscate the new transaction. Once done, BronzeGate sends the obfuscated transaction back to the Capture process which simply writes it to the trail, which shall be shipped to the replication site.

4. Analysis

In this section, we analyze the degree of data privacy, repeatability and data usability of the proposed obfuscation techniques.

4.1 Data Privacy:

Anonymization guarantees securing data 100% [1]. And hence, numerical general data obfuscated using the GT-ANeNDS and that obfuscated using a dictionary are guaranteed to secure the privacy. For identifiable numerical data, Special Function 1 obfuscates the data using two different techniques then randomly picks digits from both obfuscated values into one new output value. Without full knowledge of the original data, there is no way to find out from where each digit was picked. Thus, Data Privacy is maintained, and the proposed obfuscation techniques are immune even to Partial Attacks, in which partial knowledge about the original data and/or the obfuscation process are used to reverse engineer a portion of the original data.

4.2 Obfuscation Repeatability:

The proposed techniques guarantee repeatability, i.e., applying to the same input data results in the same obfuscated data maintaining referential integrity. In all BronzeGate techniques the randomization is dependant on the original data, that is the random seed is generated using the original data value, thus guaranteeing its repeatability.

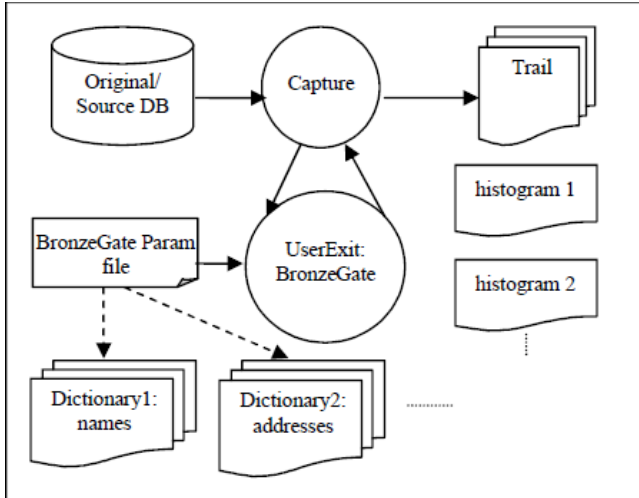


Figure 5: BronzeGate Architecture

4.3 Data Usability:

This is the hardest question to answer for numerical data since our proposed technique introduces some Anonymization. However, since we determine the number of neighbors and their distances from the origin based on the number and distribution of data values within this bucket, thus the set of neighbors should be representative enough that the anonymized data are still useable. This is further demonstrated in the experimental section.

Issues for consideration: Initial Construction of the histograms and dictionaries is the only offline process within BronzeGate. Depending on the application dynamics, this process might need to be repeated, and all database need to be rereplicated. This should be done in an efficient way, minimizing overhead and downtime. This is part of our future work.

5. Experimental Evaluation

In this section we demonstrate the BronzeGate performance to get a sense of how different techniques perform and we demonstrate the data usability.

5.1 Obfuscation Sample Results

In this experiment, an Oracle database is replicated to an MSSQL one using BronzeGate. We created one table that includes all different data types and obfuscated all fields except the notes, to identify the replicated record. Table 2 shows the first five tuples, and their obfuscated replicas. We can see from the table how identifiable numerical values (SSN and credit card) are obfuscated using the Special Function 1 into unique (i.e., identifiable) values. We updated and deleted tuples as well, and the correct replica reflected the updates, showing the repeatability of BrozeGate techniques. The table also shows for other data types how obfuscated values secure the original data.

5.2 Data Usability

In this experiment, we demonstrate the data usability of BronzeGate by applying K-mean classification algorithm, with $k=8$, using Weka Software [12] to both the original and obfuscated data using BronzeGate, and plot the results. Our workload is a dataset of protein data [13] in ARFF format.

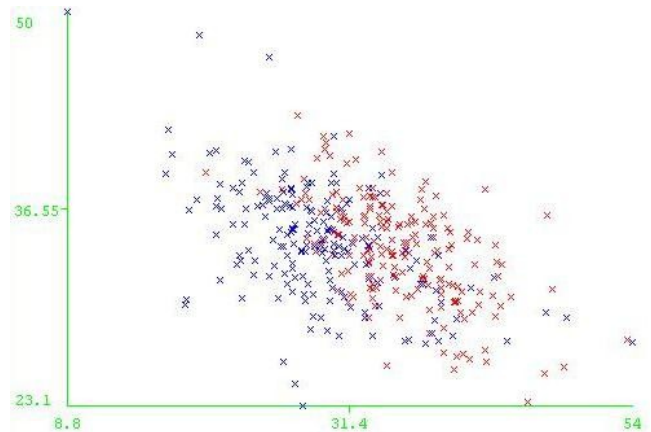


Figure 6: Data Usability: Classification of the Original Data

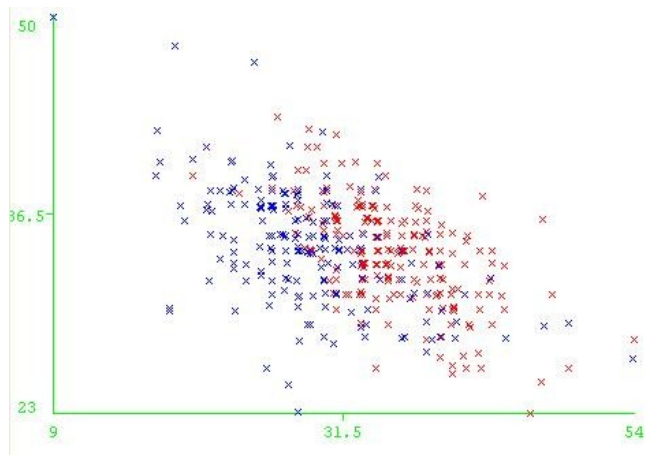


Figure 7: Data Usability: Classification of the Obfuscated Data

The results of the K-mean algorithm on original and obfuscated data are shown in Figures 6 and 7 respectively. For the data obfuscation, we applied the GT-ANeNDS with theta equal to 45 degrees, origin point was set to the min value found in the original data set, and the histogram parameters were as follows: bucket width equals to one fourth of the range of the original data set, and sub-bucket height was set to 25%, so that we have four sub-buckets in each bucket. As we can see in Figures 6 and 7, the classification results are almost exactly the same. This demonstrates the data usability of BronzeGate.

6. Related Work

In this section we provide an overview of the related work in the literature. To the best of our knowledge, BronzeGate is the first real-time transactional data obfuscation solution. The major obfuscation techniques can be grouped into: anonymization, randomization and swapping. All proposed techniques would use one or a combination of these techniques. Data Anonymization aims to secure the data by mapping many original data items into one obfuscated data item, thus totally hiding the original data. Data Randomization aims at distorting the data by introducing some noise, while Data Swapping aims at shuffling the data around. Data Anonymization generates the most secure obfuscated data, while data swapping generates the most usable obfuscated data. Our BronzeGate solution utilizes Anonymization

Table 3: Obfuscation Sample

Original Values							
	SSN	First Name	Last Name	DOB	G	Credit Card	Bank Balance
1	772877278	Paul	Dillon	Jul 23, 1980	M	0987678543543276	\$4,310.76
2	099099900	Marty	Allen	Jul 24, 1980	M	9020488205617100	\$905.17
3	293487109	Mary	Marc	Jul 25, 1980	F	1234567890123450	\$2,500.03
4	370805980	Hillary	Clinton	Jul 26, 1980	F	4134754103803630	\$270,100.32
5	832875183	Barak	Obama	Jul 27, 1980	M	1903749913701830	\$203,000.76

Obfuscated Values							
	SSN	First Name	Last Name	DOB	G	Credit Card	Bank Balance
1	116698616	PALMER	ADMON	Dec 23, 1980	M	2011116131532110	\$91.00
2	007990774	WILL	RANDALL	Jan 9, 1920	M	1399919063932230	\$671.00
3	724401663	WILL	GARFIELD	Jan 9, 1971	M	0315672200134567	\$1,683.00
4	676277247	HARLAN	RAJAN	Jan 10, 1971	F	1811011171140110	\$317,888.00
5	196990616	CECILY	ABBA	Jan 11, 1921	F	0131211312620111	\$50,816.00

to secure the data, while controlling the granularity of the Anonymization level, to control the level of usability.

In the field of data mining, many DO and privacy preserving techniques were proposed. GT-NeNDS is the current state of the art [1] as discussed in the Section [2], see [6] for an extended bibliography. In general, one can divide the contributions in privacy preserving mining techniques into distributed [8] and centralized [7], where in the latter a center that has all privacy preserved data applies a mining technique, while the former allows each party to share privacy preserved mining results. In either case, the original data can not be reverse-engineered.

Privacy preserving techniques have also dug their way to data cleaning [9] where several parties own confidential information, and at integration with other parties, data cleaning need to take place without revealing the confidential information. Obfuscation techniques were also proposed to obfuscate abstract data types [10] to transform a program, making the reverse engineering of the program a hard task, while preserving the behavior (or functionality) of the original program.

In terms of transactional data, Oracle 10g's Enterprise Manger offers Data Masking tools [11] which offers the DBA a set of out-of-the box masking techniques for different PII data, such as Credit Card numbers. Now as Oracle is acquiring GoldenGate, this facilitates the integration of BronzeGate and Oracle's masking and the usage of Oracle's dictionaries, which eliminates a non trivial task..

7. Conclusions and Future Work

In this paper, we presented BronzeGate, the GoldenGate real-time transactional data obfuscation solution. BronzeGate obfuscates the data on the fly, given meta-data, utilizing different techniques to different data types, allowing the user to overwrite the *defacto* settings, as well to control the process using input parameters. We analyzed the security, repeatability and data usability of BronzeGate and showed that data is secured 100%. In the

experimental section we demonstrated data privacy and usability, and showed how clustering mining results are very similar when applied to obfuscated data, as to original data. As future extensions, we plan to consider the process of generating the histograms and maintaining it in an efficient way.

REFERENCES

- [1] Rupa Parameswaran, "A Robust Data Obfuscation Approach for Privacy Preserving Collaborative Filtering", PhD Thesis, 2006.
- [2] <http://www.hhs.gov/ocr/privacy/>, 2009
- [3] <https://www.pcisecuritystandards.org/>, 2009
- [4] Dalenius, Tore, "Towards a methodology for statistical disclosure control", *Statistisk Tidskrift*, 5, 429-444, 1977.
- [5] Product White Paper: GoldenGate Solutions and Technology, Nov 2008.
- [6] Abdelaziz Mohaisen, "Privacy Preserving Technologies: Extended Bibliography", ETRI, Korea
- [7] Yehuda Lindell and Benny Pinkas, "Privacy Preserving Data Mining", *Journal of Cryptology*, March 2008
- [8] Mahir C. Doğanay, Thomas B. Pedersen, Yücel Saygın, Erkay Savaş and Albert Levi, "Distributed privacy preserving k-means clustering with additive secret sharing", In Proc. of PAIS Workshop, 2008
- [9] Mohamed Yakout, Mikhail Atallah and Ahmed Elmagarmid, "Efficient Private Record Linkage", ICDE 2009
- [10] Stephen Drape, "Obfuscation of Abstract Data Type", Thesis, 2004
- [11] Oracle Data Masking Pack, http://blogs.oracle.com/securityinsideout/2008/01/oracle_data_masking.html, 2009
- [12] Weka Software, <http://www.cs.waikato.ac.nz/ml/weka/>, 2009
- [13] Protein data sets, maintained by Shuiwang Ji, <http://www.public.asu.edu/~sji03/>, 2009