Data Privacy Using MASKETEER™

Sachin Lodha[1], Nikhil Patwardhan[1], Ashim Roy[1], Sharada Sundaram[1], and Dilys Thomas[2]

[1] Tata Consultancy Services (sachin.lodha, nikhil.patwardhan, ashim.roy)@tcs.com ,
[2] Stanford University (dilys, shas)@cs.stanford.edu

Abstract. Advances in storage, networks, and hardware technology have resulted in an explosion of data and given rise to multiple sources of overlapping data. This, combined with general apathy towards privacy issues while designing systems and processes, leads to frequent breaches in personal identity and data security. What makes this worse is that many of these breaches are committed by the legitimate users of the data. Major countries like the U.S., Japan, Canada, Australia and EU have come up with strict data distribution laws which demand their organizations to implement proper data security measures that respect personal privacy and prohibit dissemination of raw data outside the country.
Since companies are not able to provide real data, they often resort to completely random data. It is obvious that such a data would offer complete privacy, but would have very low utility. This has serious implications for an IT services company like Tata Consultancy Services Ltd. (TCS), since application development and testing environments rely on realistic test data to verify that the applications provide the functionality and reliability they were designed to deliver. It is always desirable that the test data is *similar* to, if not the same as, the production data. Hence, deploying proven tools that make de-identifying production data easy, meaningful and cost-effective is essential.
Data masking methods came into existence to permit the legitimate use of data and avoid misuse. In this paper, we consider various such techniques to come up with a comprehensive solution for data privacy requirements. We present a detailed methodology and solutions for enterprise-wide masking. We also present the data masking product MASKETEER™ , developed at TCS, which implements these techniques for providing maximum privacy for data while maintaining good utility.

1 Introduction

Suppose that a large corporation wants TCS to develop an application that is going to use a lot of its sensitive data. Its main concern is about the security and confidentiality of its data, for it needs to protect the interests of its clients and partners. TCS presents its security practices and security audit certificates. However, the client corporations senior management is not convinced. They fear that an application developer with access to this data may use it maliciously. They suggest that the software development vendor should use some fictitious data during the application development cycle.

Application development and testing environments in TCS rely on realistic test data to verify that the applications provide the functionality and reliability they were designed to deliver. So it is desirable that the test data is similar to, if not same as, the

production data. This leads to a deadlock situation, where in TCS needs the original data to develop a good product while owing to the privacy concerns of its clients the large corporation cannot give the data to TCS. One solution is to change or mask the production data when migrating it to the test environment. Masking the production data is *the process of systematically removing or transforming data elements that could be used to gain additional knowledge about the sensitive information*. That is the data is as realistic as possible for the test environment and also as fake as possible to respect privacy.

The objective of data masking is to maximize data utility in such a way that the masked data should have the same characteristics as the original data and at the same time minimize disclosure risks, that is, reduce the ability to identify an individual and reduce the ability to predict the value of confidential attributes. Note that there is a natural trade-off between the data privacy and its utility. An unmasked, original data would naturally have the highest data utility, but no privacy. On the other hand, randomly generated data would guarantee very high privacy, but almost no utility. However, if done properly, data masking should combine the best of both the worlds, that is, high data utility as well as high data privacy.

It is important to note that the results of the data transformation have to be appropriate in the context of the application. Data utility is dependent on the use of the masked data. For example for testing purposes larger volumes of syntactically correct data is needed with all the outliers. While for data mining or knowledge discovery the data needs to be as realistic as possible but completely de-identified.

The need for data masking is, in fact, ubiquitous. The contract software development is just one such striking example. Many useful properties can be extracted from the data even if it is masked. Data Mining is one such example, where certain statistical properties of the data set can be retained even by destroying their association with the original data. Data masking plays a key role when a certain version of privately held data has to be made public. Here goal is to keep the identities of the individuals who are the subjects of the data secret, and yet allow the legitimate users to make perfect use of the released data. This problem is very common in the health sector. Government agencies would need to solve the same problem while releasing census data. One more scenario where data masking is of crucial significance is location-based applications. Here the infrastructure should allow mobile users to avail all possible location-based information services without endangering their location privacy.

2 Masking Approaches

Over the years, statisticians, cryptographers and computer scientists have developed many models and techniques to address the trade-off between data privacy and its utility. We present here techniques that are robust, practical, and have simple quantification of privacy/utility. We explain them using Figures 1 and 2.

Randomization: In this approach, a data-element is replaced by a randomly chosen value from a given range or a dataset. The **Name** column of Figure 1 is replaced by randomly chosen names from a dataset of `English Names` in Figure 2. This technique provides strong identity protection.

SSN	Name	Gender	Age	Zipcode	Balance
101	Alice	F	31	94305	100
102	Bob	M	31	94308	24
103	Carol	F	32	94308	35
104	David	M	22	94125	85
105	Evelyn	F	34	90428	12
106	Frank	M	18	94308	73
107	George	M	35	92405	57

Fig. 1. Original Database table

SSN	Name	Gender	Age	Zipcode	Balance
501	Jane	F	31	9430*	110
438	Kurt	M	31	9430*	30
107	Lance	M	31	9430*	45
745	Molly	F	20	94***	75
885	Nancy	F	34	9****	25
990	Oscar	M	20	94***	60
210	Philip	M	34	9****	52

Fig. 2. Masked Database table

Hashing and Encryption: In this approach, the data-element X is replaced by its image $h(X)$ where h is a suitable hash function. Ideally, one would want the hash function h to be collision-free, non-idempotent and one-way. Unfortunately no such h is provably known. But in practice, `MD5`, `SHA-1`, or Discrete Log based functions are useful. In the above example, encryption is applied to the **SSN** column. Encryption techniques are often efficient, but they provide low data utility by destroying semantics readily.

Shuffling: Shuffling randomly permutes the data-elements in a column. Thus, it can easily destroy relations between the columns. Shuffling is applied to the **Gender** column in the above example.

Perturbation: Another popular approach is to use perturbation techniques in order to hide the exact values, for example, adding noise to data and its numerous improvements. Here it is possible to capture the richness of data, say, with the covariance matrices . A simple perturbation technique could be addition of Gaussian noise to the input data. Let X be the input column. Then, the resultant Y would be $Y = X + e$, here e is the Gaussian noise taken from a standard distribution. Perturbation is applied to the **Balance** column in the above example.

Perturbation techniques, capable of providing high data utility and low disclosure risk, may require some pre-processing of the data to yield parameter values. Otherwise, they are fairly efficient. They are not very suitable if one wants to draw inferences with 100% confidence.

k-Anonymity: De-identifying the data by masking key attributes like **SSN** and **Name** may not protect identities since linking such a masked database with a publicly available

database on non-key attributes like **Gender**, **Date of Birth** and **Zipcode** can uniquely identify an individual [10].

A table provides k-anonymity [10] if any attempt to link the identifying columns by external joins results in k or more matches. It means that each row in the table is forced to be same as at least $k-1$ other rows in the potentially identifying attributes. Thus, identification of an individual by external join is with a probability of at most $1/k$. k-Anonymity is achieved by blocking all the dissimilar values (suppression) or by replacing them with a less specific common consistent value (generalization). Hence k-anonymity is a trade off between data utility and privacy. A higher value of k enforces a stricter privacy but more data loss. It is important to note that the data loss happens mainly for the potentially identifying attributes only, and the sensitive information (for example, say, medical condition of patients or cash balance) may appear as it is.

In the above example, we have applied 2-Anonymity to **Age** and **Zipcode**. The first three, the 4th and 6th, and the 5th and 7th rows in Figure 2 are identical in these two columns. Non identical values in **Age** are replaced by their average, while those in **Zipcode** are substituted by *. Hence anybody trying to link some known **Age** and **Zipcode** values with those in this table to find the **Balance** value would essentially be confused with *two* balance values. Thus, we are guaranteed 50% privacy.

k-Anonymity provides high data utility since it generalizes or suppresses only the quasi-identifiers. It also quantifies the identity disclosure risk at $1/k$. But the optimal k-anonymity is known to be NP-hard [2]. The known algorithms are either $O(k)$ approximations [2] or super-linear [1] or require time exponential in the number of columns [7] thus making them inefficient or expensive. We have designed and implemented a single pass algorithm that scales linearly in the number of columns [8] in our product.

A comparative overview of the different techniques is given in Figure 3. It is clear from this table that no single technique by itself can provide low disclosure risk, high data utility and work for high data volumes. But good news is that these techniques seem to complement each other. So their right combination may generate *good* data for us. A recent Forrester report [13] also advocates the same.

Technique	**Data Utility**	**Identity Disclosure Risk**	**Value Disclosure Risk**	**Scalability**
Randomization	Low	**Low**	**Low**	**High**
Encryption	Low	Medium	Medium	**High**
Shuffling	Medium	**Low**	High	Medium
Perturbation	Medium	**Low**	Medium	Medium
k-Anonymity	**High**	**Low**	High	Low

Fig. 3. Techniques Overview

3 Masking Challenges

The simple techniques described above are privacy-preserving and amenable to efficient implementations that scale well. Also appropriate choice of parameters for these

techniques can help us retain much of the characteristics and patterns from the original data providing sound results. But there are other database constraints that limit the ability of masking. For example theoretically k-anonymity kind of solution would provably provide better privacy [10, 8] and preserve some vital statistical properties, but if any one of the columns is declared as unique then a method like k-anonymity cannot be applied at all. This is where theory parts ways with practice and we have to consider the real world restrictions and accommodate them. They bring in serious engineering challenges. The database related constraints can be broadly classified as

Syntactic Constraints: The masked values should be within the limits specified in the database schema.

Uniqueness Constraints: For attributes that are marked as unique or as primary keys of the table, the masked values should be unique.

Relational Integrity: In order to support RDBMS, it is important to make sure that relational integrity constraints are satisfied, that is, the masked data-elements are propagated from the parent table to child tables.

Business Constraints: Much of the known business logic is encoded in company databases by linking multiple columns through some arithmetical or logical operations. The masked data should also satisfy these constraints.

Semantic Constraints: The data values in real life could be clustered around a certain value or have some distinctive association with values in other columns. From the masked value more information about the original values should not be obtained due to some uncharacteristic organization of the data.

4 Information Retrieval from Masked Data

Masking data is important for other purposes like outsourced data mining and remote analysis of data [10, 7, 9]. It is also useful for sanitizing data before putting it together from multiple sources [3, 11, 4]. Data put together from multiple sources could be **horizontally partitioned** or **vertically partitioned**. In case of horizontal partitioning the different data sources have data conforming to the same schema but contribute data about different rows or individuals. This allows more information to be learned from the combined data and also allows the discovery of rare facts as there is more data. In the case of vertical partitioning the different data sources provide data about the same rows or individuals. However the different sources provide different aspects, features or columns corresponding to the rows or individuals. This allows the inferring of knowledge correlating different columns, which may not be possible at any single data source.

Extracting information and learning functions from data distributed on multiple sources has been well studied for two decades. The area of secure multiparty computation [12, 5] has allowed the computation of an arbitrary function expressed as a circuit on top of the original data sources. This can afford the sources the strongest notion of privacy, semantic security [6]. However more recent research tries to give the data sources weaker notions of privacy [11, 3, 4] while at the same time provide techniques for easy computation of a large class of useful functions on these distributed data sources.

It is important during data sanitization to retain as many properties of the data so that the data mining algorithm will be able to learn maximum amount of information from the database. For tools like MASKETEER™ we can come up with principles to decide the sanitization to be applied to different columns before giving it out for knowledge discovery. For identifier columns like SSN, drivers license number, voter number etc. randomization or hashing are popular techniques as they provide good data privacy. For other columns which are also available at other sources and are susceptible to join attacks (called quasi-identifiers in contemporary research), like date of birth, gender and zipcode k-anonymity [10, 7, 8] is the technique to be applied. Sensitive columns which are only available at the data source can be subjected to l-diversity [9] transformation. Perturbation can be applied to numeric columns, and shuffling can be applied to categorical columns.

5 Masking Best Practices

Data masking is not just applying a standard algorithm to all information. The choice of the masking algorithm and its parameters depend on the desired application of masked data, the database constraints and also the actual values present in the production data. There is no single right solution but a slew of things need to be taken care before sending the masked data out. After interacting with many TCS clients from banking, finance, and health sectors, as well as detailed market studies, we recommend these guidelines for achieving the best results:

1. **Analysis** It is important to understand the complexity of the application and data environment. This would help to identify sensitive data at the bare minimum (as opposed to all), understand data relationships and constraints defined both in data as well as in application. Sufficient rigor is needed to understand the flow of data across applications, various assumptions, both at the application and inter-application level. Even all the sources of the data, their flow and their copies needs to be known to avoid unwarranted data leakage. A good knowledge of the applicable regulatory laws is also essential for their compliance and coming up with a masking strategy.
2. **Setup** This part is crucial in the sense that the person, who has the authority to access the original data needs to setup the whole masking strategy of selecting the sensitive data, applying the right masking technique and taking into consideration the risks involved in identifying the data back. We suggest a few guidelines for using the algorithms described above.

   Randomization Useful for performance testing
   - Data is effectively useless
   - Privacy is maximum
   - Domain for randomization needs to carefully defined from some external source.

   Encryption and Hashing Useful for relinking back to original data
   - Semantic information about the data is lost to do any meaningful analysis.
   - Encryption and hashing algorithms should be coded with great care to avoid any possible error.

   Shuffling Useful for testing as outliers are maintained

- Bad for non-uniformly distributed data.
- Data associations with other columns are lost.
- Original data is in the clear.

Perturbation Testing statistical analysis and data mining
- The amount of noise depends on the distribution of the data.
- Lot of advance research exist about such techniques and their guarantees.
- Applicable mainly to numerical data.

k-Anonymity Aggregate analysis and data mining
- Good against a weak attacker model
- Not good for unique and sparse data values.

All the data integrity constraints need to be in place for correct masking and only then the masking should be carried out. If the database size is huge then sampling would be a good idea to reduce processing time. If masking is a regular job then based on the strategy a schedule of the test bed creation can programmed.

3. **Masking** The actual masking of the data could be a time consuming process depending on the amount of data to be masked. There needs to be a way to recover from failures if any. At the end of masking, user must verify the quality of masking and identify any unintended variation on the statistical measure or amiss from the specification.
4. **Upload and Final Review** Once the masking is done, there should be some easy way to transfer the masked data to various target systems. A thorough check to ensure the inter-relationships and constraints of data are maintained is necessary so that the data is acceptable the to various applications using it.

6 Conclusion

MASKETEER™ supports most of the RDBMS using, say, JDBC drivers for connectivity. For example, it supports DB2, MS Access, Oracle, MS SQL Server, Sybase, Informix, MySQL, VSAM, TeraData, Advantage, DB2 on AS 400, PostgreSQL, IMX, XLS and also flat files. Its extensible framework allows users to plug-in other databases as well. MASKETEER™ is available on Mainframe, Windows, Linux ,Unix and Mac. The top sixteen customers of MASKETEER™ , including customers in different industry units like BFS, Insurance, Telecom and HiTech have masked totally hundreds of Gigabytes of data, stored in thousands of tables, spanning tens of databases. [1] This simple, intuitive, versatile, user-friendly and easy to use tool has served as a good business enabler for TCS and has won 2010 Golden Peacock Award for Innovation Product.

Acknowledgement We will like to thank everyone who has helped in the making of MASKETEER™ , namely, Raghava Annaparthi, Yogesh Athavale, Sitaram B, Nandita Babu, Arpita Baheria, Aditya Bahulekar, Vijayanand Banahatti, Munesh Bandaru, Joel C, Siddhartha Chatterjee, Amitesh Chellaramani, Prasenjit Das, Anshula Dhar, Rahul

[1] For more information and also evaluation copies of MASKETEER™ , please feel free to write to global.toolscare@tcs.com. Visit our website at: http://www.tcs.com/offerings/technology-products/masketeer-data-privacy-solutions/Pages/default.aspx

Ghodeswar, Hema Gopalan, Chandrashekhar Hire, Srinivasan Iyengar, Sumit Johri, Kalaiselvan K, Rahul Kanwar, Vijaya Kedar, Aniket Kulkarni, Amol Limaye, Sanooja M, Snigdha Nayak, Jayant Patil, Ellora Praharaj, Ravi Rajamony, Chandrama Ramkumar, Sasanka Roy, Barath S, Rupali Sagade, Aparna Sarnobat, Vinayak Sharma, Manish Shukla, Lovekesh Verma, Kumar Vidhani, Avinash Wagh and Leena Walavalkar. Special thanks to Professor Mathai Joseph ((now retired) Executive Director of TRDDC), K Ananth Krishnan (Chief Technology Officer, TCS), Professor Harrick Vin (Head of Systems Research Lab at TRDDC) and Arun Bahulkar (Head of Software Engineering Lab), Vijayalakshmi Gopal (Head of TCS Tools Group) and Dr. Santosh Mohanty (Head of Component Engineering Group) for their continued guidance, support, and encouragement for this activity. We would like to thank Professors John Mitchell, Rajeev Motwani, Hector Garcia-Molina, Dan Boneh and Jeffrey Ullman for leading the privacy research group at Stanford.

References

1. G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 153–162, 2006.
2. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proceedings of the International Conference on Database Theory*, pages 246–258, 2005.
3. R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 251–262, 2005.
4. C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Proc. CRYPTO*, pages 134–138, 2004.
5. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game – a completeness theorem for protocols with a honest majority. In *Proceedings of the 1987 Annual ACM Symp. on Theory of Computing*, pages 218–229, 1987.
6. S. Goldwasser and S. Micali. Probablistic encryption. In *Journal of computer security, Vol. 28*, pages 270–299, 1984.
7. K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60, 2005.
8. S. Lodha and D. Thomas. Probabilistic anonymity. *Privacy, Security and Trust in KDD, PinKDD, with SIGKDD*, pages 56–79, August 2007.
9. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the International Conference on Data Engineering*, page 24, 2006.
10. L. Sweeney. *k*-Anonymity: A model for preserving privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.
11. J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 2002 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644.
12. A. Yao. How to generate and exchange secrets. In *Proceedings of the 1986 Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.
13. N. Yuhanna. Protecting private data with data masking: Technology can help meet certain compliance and outsourcing requirements. Technical report, Forrester Research, 2006.