# Secret Sharing vs. Encryption-based Techniques For Privacy Preserving Data Mining [1]

Thomas B. Pedersen, Yücel Saygın, Erkay Savaş

Faculty of Engineering and Natural Sciences
Sabanci University, Istanbul, TURKEY

**Abstract**. Privacy preserving querying and data publishing has been studied in the context of statistical databases and statistical disclosure control. Recently, large-scale data collection and integration efforts increased privacy concerns which motivated data mining researchers to investigate privacy implications of data mining and how data mining can be performed without violating privacy. In this paper, we first provide an overview of privacy preserving data mining focusing on distributed data sources, then we compare two technologies used in privacy preserving data mining. The first technology is encryption-based, and it is used in earlier approaches. The second technology is secret-sharing which is recently being considered as a more efficient approach.

## 1 Introduction

With the popularity of Internet, it is now extremely easy to collect person-specific data which can also be linked to other data sets. Ubiquitous devices such as RFID tags and readers and GPS equipped mobile phones increased the privacy concerns as well, since they made it possible to collect location information about people. But the turning point was the 9-11 events after which US government increased its nation-wide data collection and integration efforts in the name of "fight against terrorism." In fact, some of the largest airline companies in the US, including American, United and Northwest, turned over millions of passenger records to the FBI according to NY Times. Such scandals proved that privacy risks are real. Privacy breaches could also be accidental as in the case of the AOL scandal. AOL released the "de-identified" search logs of its 650.000 customers over a 3 month period in August 2006. AOL realized its mistake and removed the data, but is was already downloaded by many users, and in fact an individual was being identified from her query logs.

Data mining is motivated by the large-scale data collection efforts by companies and government organizations with the aim of turning massive amounts of raw

[1]This work was partially funded by the Information Society Technologies Programme of the European Commission, Future and Emerging Technologies under IST-014915 GeoPKDD project.

data into useful information. Machine learning, Artificial Intelligence, Statistics, and Databases are utilized in data mining in order to come up with data-centric techniques for extracting models from massive data collections. The extracted models could be in many forms, such as rules, patterns, or decision trees. Most of the profitable applications of data mining concerns humans. Therefore a considerable proportion of the collected data is about people and their activities. This is why data mining and privacy discussions are inseparable now. In fact some data mining projects were not funded due to privacy concerns. According to Computer World [21], "The chairman of the House Committee on Homeland Security has asked Department of Homeland Security Secretary Michael Chertoff to provide a detailed listing of all IT programs that have been canceled, discontinued or modified because of privacy concerns." In response to such privacy concerns, data mining researchers started working on methods for preserving privacy when doing data mining. Techniques were developed for different data mining models, starting from classification models, then association rules and clustering for distributed scenarios.

The research efforts on privacy preserving data mining at European level were supported by two large scale projects funded by Future and Emerging Technologies of Information Society Technologies under the 6th Framework. One of the projects is Geographic Privacy-aware Knowledge Discovery and Delivery (GeoPKDD), and the other is Knowledge Discovery in Ubiquitous Computing (KdUbiq). These projects have different purposes but they are both concentrated on new data mining technologies and their privacy implications. GeoPKDD is a research project concentrating on spatio-temporal knowledge discovery, and privacy issues in spatio-temporal knowledge discovery. KdUbiq aims at creating a community in the area of ubiquitous knowledge discovery which will define the area and research directions. One of the working groups is privacy and security in ubiquitous computing.

In this paper we are going to concentrate on privacy preserving data mining in distributed environments and discuss two classes of techniques, namely the encryption based and recently introduced secret sharing based techniques for privacy preserving data mining.

## 2 Overview and State-of-the-art

In this section we summarize the pioneering work on privacy preserving data mining which shaped the research in this field. Two concurrent papers published in 2000 by researchers from different groups used the title "Privacy Preserving Data Mining"[3, 14]. Although the titles were the same, their approach and problem setting was different. The authors of [3] assumed that the data mining effort will be outsourced to a third party. Before the data could be handed over to a third party the confidential values in the database, such as the salary of employees, needed to be perturbed in a way that the original probability distribution could be estimated

from the perturbed data but not the original data values. This way, a decision tree can still be constructed from the perturbed data within a certain error margin that the authors approve. The authors in [14] assume that there are two parties with private data sources who would like to do data mining without seeing each other's data and propose cryptographic techniques to achieve that. They also demonstrated their approach on decision tree construction.

In distributed data mining one or more *data holders* provide the input data for the data mining, while one or more *data miners* cooperate in performing the data mining. A simple way to perform distributed data mining is to send all data to one data miner, and let the data miner perform the data mining. If the data contains private information such an approach clearly violates privacy, since the data miner will have full access to all data. In such a distributed environment it is thus not enough to ensure that the result of the data mining preserves privacy, we must also guarantee that privacy is not breached during the computation itself. Kantarcioglu and Clifton developed protocols to privately mine for association rules in a distributed environment[12]. The authors considered a case in which there are multiple parties that have their own confidential local databases that they do not want to share with others. The assumption is that the data is distributed horizontally, i.e., the database schema is the same for all the parties. The individual association rules together and their statistical properties were assumed to be confidential. Another assumption is that the involved parties are honest but curious, i.e., they follow the protocol but they would like to get as much information as possible from the data they receive. Under these assumptions, secure multi-part computation base protocols were employed based on commutative encryption schemes to make sure that the confidential association rules are circulated among the participating companies in encrypted form. The resulting global association rules can then be obtained in a private manner without each company knowing which rule belongs to which local database.

When data is going to be published, or handed over to a third party, it needs to be sanitized by removing sensitive information. This sensitive information could be some data values or it could be in the form of data mining models. In [2], the authors propose an approach for privacy preserving data mining which maps the original data set into a new anonymized data set preserving the correlations among the different dimensions.

Security of random perturbation methods against partial disclosure through successive querying of the database by snoopers is studied in [17]. The effect of high dimensionality in randomisation was studied by Aggarwal in [1]. In the work by Liu *et al.* the authors point out that perturbation techniques which preserve distance between data objects can be attacked if the attacker knows a small set of data selected according to the same probability distribution as the original data set[15, 16]. The attack applies principal component analysis to the perturbed data and tries to

fit it to the known data set. Liu *et al.* also propose an alternative transformation where the objects in the original data set are projected onto a subspace in a way that distance is preserved with high probability. They point out that the alternative approach is secure against the identified attack, but may not be secure against other attacks.

## 3 Encryption-based Techniques

Research in privacy preserving data mining started after 2000, but the cryptographic background dates back to Yao's definition and solution to the "millionaires problem" in 1982[22]. In Yao's millionaires problem two millionaires want to find out who is richer, but without revealing their wealth to the each other. In fact, the ability to compare numerical data is crucial in most data mining tasks. Yao's work initiated research in *secure multi-party computation* which is the study of the class of functions which two or more players can securely compute on their joint inputs. This is done in a way that nothing but the final result of the computation will be revealed to the parties. In particular, no party will know the inputs of the other parties. Yao later on demonstrated that any problem which can be described by a polynomial size boolean circuit of logarithmic depth can be solved securely[23]. Today we know that any computation which can be done in polynomial time by one party can be done securely by multiple parties[5]. The only ingredient needed in these generic protocols is encryption.

An important issue in secure multi-party computation is the definition of security. What does it mean that "the protocol for computing function $f$ does not reveal too much"? Intuitively we would like our protocols to be as if all players send their inputs to an *honest third party*, which performs the computation and returns the results to the players. This perfect protocol is clearly secure, since no player sees anything else than its own inputs and outputs (in multi-party computation we do not address the privacy issues of the input itself). A formalisation of this idea is the standard definition of secure computation used today[10]. The definition requires the existence of a *simulator* which can generate the state of any (possibly dishonest) party at each step of the protocol when given the inputs and outputs of the party in the protocol. The simulated state should be such that it is not computationally feasible to tell the difference between the simulated state of the dishonest participant, and the state of the participant in a real invocation of the protocol.

Some of the well-known *public-key encryption schemes* are RSA and ElGamal. RSA encrypts messages of approximately 1024 bits in ciphertexts of 1024 bits. ElGamal is an elliptic curve based encryption which can handle messages (typically around 160 bits) that are much smaller than what RSA can handle. Public key encryption schemes are easier to use and administrate, but are slower than the so called *symmetric key encryption* schemes. DES and AES are well known symmetric

key encryption schemes. They encrypt messages of 64 and 128 bits respectively, and generate ciphertexts of the same length.

### 3.1 Circuit Evaluation

Many of the protocols based on encryption use the idea introduced by Yao[23]. In Yao's protocol one of the parties compute a scrambled version of a boolean circuit for evaluating the desired function. The scrambled circuit consists of encryptions of all possible bit values on all possible wires in the circuit. The number of encryptions is approximately $4m$, where $m$ is the number of gates in the circuit. The encryptions can be symmetric key encryption, which has a typical ciphertext-length of 64 bits. The scrambled circuit is sent to the other party, which can then evaluate the circuit to get the final result. These approaches are, in general, expensive since they require complicated encryptions for each individual bit.

Many privacy preserving data mining protocols use the idea of scrambled circuits, but in order to limit the overhead of scrambled circuits, they only use scrambled circuits as sub-protocols to compute certain simple functions[14, 20].

### 3.2 Homomorphic Encryption

A powerful tool in computing a wide range of functions with computational security is *homomorphic encryption*. With homomorphic encryption we can avoid the bit-wise encryption from the scrambled circuits described in Section 3.1. Homomorphic encryption schemes are a special class of public key encryption schemes. The first homomorphic cryptosystem, called the Goldwasser-Micali (GM) cryptosystem, was proposed in 1984[11]. Due to its prohibitive message expansion during encryption (i.e. each bit of plaintext is encrypted as a ciphertext of eat least 1024 bits), it is not practical for data mining applications. The natural extension of the GM cryptosystem is the Benaloh cryptosystem [21], which allows the encryption of larger block sizes at a time. Although the message expansion is not as bad as in the GM cryptosystem, it is still not suitable for data mining applications. Furthermore, the fact that the decryption is based on exhaustive search over all possible plain-texts also makes the Benaloh cryptosystem unpractical for privacy preserving data mining. A more recent scheme is the Paillier cryptosystem [18], which avoids many of the drawbacks of the earlier homomorphic cryptosystems. The Paillier cryptosystem provides fast encryption and decryption algorithms, and it encrypts 1024-bit messages in ciphertexts of at least 2048-bits, which is reasonable if we work with large plaintexts.

Homomorphic encryption enables us to compute certain functions more efficiently compared to scrambled circuits. The authors of [9] use homomorphic encryption for computing secure scalar products used in privacy preserving data mining. The protocol is shown in Fig. 1, where two players, A and B, compute the scalar product of vectors $\overline{v} = (v_1, \ldots, v_d)$, and $\overline{w} = (w_1, \ldots, w_d)$, such that only B learns the scalar

product, and A learns nothing at all.

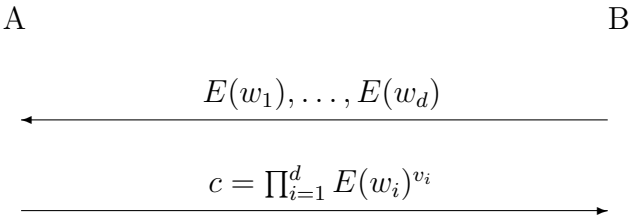


Figure 1: Computationally secure scalar product protocol ($D(c) = \overline{v} \cdot \overline{w}$).

## 4 Secret Sharing

Secret sharing was introduced independently by Shamir[19] and Blakley[7] in 1979. The idea is that one party has a secret which it distributes among $n$ other parties in a way that none of the $n$ parties alone can recover the secret. As a matter of fact the secret is shared in a way that the information of at least $t$ of the $n$ parties is needed to recover the secret, where $t$ is a predefined threshold. Any attempt by less than $t$ parties to recover the secret will fail and they will not learn *anything* about the secret.

A $(t, n)$ *secret sharing scheme* is a set of two functions $S$ and $R$. The function $S$ is a *sharing function* and takes a *secret* $s$ as input and creates $n$ *secret shares*: $S(s) = (s_1, \ldots, s_n)$. The two functions are selected in a way that, for any set $I \subseteq \{1, \ldots, n\}$ of $t$ indices $R(I, s_{I_1}, \ldots, s_{I_t}) = s$. Furthermore we require that it is *impossible* to recover $s$ from a set of $t - 1$ secret shares. A secret sharing scheme is *additively homomorphic* if $R(I, s_{I_1} + s'_{I_1}, \ldots, s_{I_t} + s'_{I_t}) = s + s'$.

A very simple $(n, n)$ additive secret sharing scheme is $S(s) = (r_1, \ldots, r_{n-1}, r)$, where $r_i$ is random for $i \in \{1, \ldots, n-1\}$, and $r = s - \sum_{i=1}^{n-1} r_i$. To recover $s$ all secret shares are added: $s = r + \sum_{i=1}^{n-1} r_i$. If even one secret share is missing nothing is known about $s$.

Shamir secret sharing was used by Ben-Or *et al.* [6] in 1988 to show that any function of $n$ inputs can be computed by $n$ parties such that no coalition of less than $n/3$ of the parties can gain *any* additional information about the honest parties inputs (even if they do not behave according to the prescribed algorithm). If we assume that all parties behave *semi-honestly* (i.e. they follow the protocol), then no coalition of less than $n/2$ of the parties can gain any information about the inputs of the honest parties. The protocol uses the additively homomorphic property of Shamir secret sharing. The idea is that addition and multiplication together is enough to evaluate any function (in particular addition and multiplication over $Z_2$ is a universal set of boolean operations). The bottleneck of the algorithm in [6] is multiplication. Since Shamir secret sharing is not multiplicatively homomorphic, in

order to perform a multiplication, a special "degree reduction step" has to be performed. This degree reduction requires that all parties secret share a new number (a total of $n^2$ new messages for each multiplication). For most data-mining applications this degree reduction step is too costly, since a vast number of multiplications is common. Another limitation of the generic multi party computation based on secret sharing is when some parties may behave dishonestly. They may try to gain extra information by deviating from the prescribed protocol. To avoid this, a special variant of secret sharing is used. This variant, called *verifiable secret sharing* adds extra information to each secret share, such that any set of players, at any time in the protocol, can verify that the shares they have are consistent. Both the extra information in the secret shares, and the interaction required to verify a secret sharing adds extra communication overhead to the protocols.

### 4.1 The Coopetative Model

Data holders that participate in distributed data mining have naturally an interest in the result of the data mining. They are, however, understandably reluctant to share their private data with others to either protect their interests or meet privacy requirements imposed by authorities and/or clients. Data holders, in other words, are ready to *cooperate* with each other to extract useful information from combined data while *competition* among them dictates that individual data is not revealed to others.

The term *coopetation* is used in economics to refer to cooperation between competing entities to improve the overall value of their market. This is quite similar to the distributed data mining scenario where data holders behave with similar motivations. In the coopetative model data holders provide inputs to a relatively small set of data mining servers or so called third parties, which are assumed semi-honest (i.e. they are honest but curious; they follow the protocol steps, but are interested in any leaked information). Some of the data holders can actively participate in the distributed data mining playing the role of third parties. The non-collusion property must be satisfied by certain sets of third parties. At the end of the protocol the data miner, which can be either a separate entity or one of the data holders, will have the outcome as an output.

Some of the benefits of the coopetative model are:

- Very efficient data mining protocols can be constructed.
- The major workload can be put on a small set of dedicated servers which are better protected and regulated.
- Only these small sets of servers need to posses the necessary hardware, software and know-how to perform data mining.

- Encryption is avoided, thus key distribution and other problems related with encryption are no longer an issue.

The basic version of the coopetative model [13] requires two dedicated third parties and a miner. Data holders secret shares their data and send each share to one of the third parties. For sake of simplicity we can assume that the private input of each data holder is an integer $x$ and the data holder creates two shares $r$ and $x - r$ where $r$ is a randomly selected integer. The share $r$ is sent to the first third party and the share $x - r$ is sent to the other. Clearly both shares are random when observed alone and no single entity (adversary, third party, or miner) can obtain any information about the private input $x$. The private input can only be revealed when two shares are put together, which never happens in the coopetative model.

The third parties work on the individual shares and compute algebraic operations such as numerical difference and comparison on the shares, which are the fundamental operations in many data mining applications (e.g. constructing decision trees, association rule mining and clustering). The result of these operations are the shares of the final outcome of the computation, which can be obtained only by the data miner.

In order for the third parties to work on shares, they need to employ secret sharing schemes which are homomorphic with respect to the operations they perform. For instance, additive secret sharing described above is homomorphic with respect to addition (and subtraction): adding shares pairwise gives an additive sharing of the sum of the secrets. Therefore, the additive secret sharing scheme can directly be used in numerical difference operations in clustering algorithms.

## 5 Discussion and Comparison

When comparing the efficiency of two alternative protocols for a specific task there are three factors to consider: the computation cost, communication cost, and the number of rounds in the protocols. We will compare encryption-based techniques and secret sharing with respect to these three factors in the following.

Public key encryption schemes are (by definition) based on computationally difficult problems, and thus require expensive operations such as modular exponentiation of large numbers (in the order of 1000 bits). In contrast it is very efficient to compute secret shares when using e.g. Shamir secret sharing or the simple additive secret sharing described in Section 4. Sharing a secret with Shamir secret sharing consists in choosing a random polynomial and evaluating it in $n$ points. The polynomial is chosen over the same field as the secret, which means that usually all computations are done with ordinary integers.

As mentioned in Section 3 public key encryption schemes create ciphertexts of at least 1024 bits (with the exception of Elliptic curve based encryption schemes). If we want to use the homomorphic properties of an encryption scheme we have to

encrypt each input in it's own ciphertext. Often this will mean that we encrypt 32-bit numbers in 1024 bits (giving an overhead of 32). If we use circuit evaluation techniques we are forced to encrypt each bit as at least 160 bits if we use elliptic curve cryptography. In contrast, secret sharing creates $n$ shares of each input, where each share is of the same size as the secret. We thus always have an overhead of $n$. Note, however, that some secret sharing schemes, like the Asmuth Bloom scheme[4], create shares which are larger than the secret.

If Shamir secret sharing is used as described in Section 4 each multiplication requires that each pair of parties exchange information. Having to wait for the transmission of these messages at each multiplication clearly slows down a protocol, in comparison Yao's circuit evaluation only requires one round of communication. Some work has been done to minimise the number of rounds needed by secret sharing based techniques[8], though they do not give constant round complexity as in the case of Yao's protocol. It is still an open problem to fully classify the problems which can be solved with a constant number of rounds with unconditional security.

We should note that not all problems can be solved with unconditional security. A very important fact, from a data mining point of view, is that unconditionally secure scalar products between two parties is impossible. Any two-party data mining algorithm which applies scalar products (between secret vectors held by the two parties) has to rely on either encryption based techniques, or external parties.

## 6 Conclusion

Privacy preserving data mining is an ongoing research area and there are a lot of issues that needs to be addressed. First of all, the databases that are collected for mining are huge, and scalable techniques for privacy preserving data mining are needed to handle these data sources. Secret sharing based methods can be considered one step forward in scalable privacy preserving data mining. The degree of data distribution could also be a problem when we consider a grid, peer-to-peer, or ubiquitous computing environments. Techniques which minimize the amount of computation and data transfer are needed in highly distributed environments. New data types such as spatio-temporal data collected by location-based services, and other mobile service providers pose new types of threats to privacy, and existing techniques for privacy preserving data mining may not be adequate to handle these types of data.

## References

[1] C. C. Aggarwal. On randomization, public information and the curse of dimensionality. In *ICDE*, pages 136–145, 2007.

[2] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving

data mining. In *EDBT*, pages 183–199, 2004.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 439–450. ACM, 2000.

[4] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2):208–210, March 1983.

[5] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 503–513. ACM, 1990.

[6] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the nineteenth annual ACM conference on Theory of computing (STOC)*, pages 1–10. ACM Press, 1988.

[7] G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of AFIPS 1979 National Computer Conference*, pages 313–317, June 1979.

[8] R. Cramer and I. Damgård. Secure distributed linear algebra in a constant number of rounds. In *Advances in Cryptology - CRYPTO 2001: 21st Annual International Cryptology Conference*, pages 119–138, 2001.

[9] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Information Security and Cryptology — ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 104–120. Springer-Verlag, 2005.

[10] O. Goldreich. *The Foundations of Cryptography — Volume 2, Basic Applications*. Cambridge University Press, May 2004.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *J. COMP. SYST. SCI.*, 28(2):270–299, March 1984.

[12] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.

[13] S. V. Kaya, T. B. Pedersen, E. Savaş, and Y. Saygin. Efficient privacy preserving distributed clustering based on secret sharing. In *LNAI 4819 PAKDD 2007*, pages 280–291. Springer, 2007.

[14] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology (CRYPTO'00)*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–53. Springer-Verlag, 2000.

[15] K. Liu, C. Giannella, and H. Kargupta. An attacker's view of distance preserving maps for privacy preserving data mining. In *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4213 of *Lecture Notes in Computer Science*, pages 297–308. Springer, 2006.

[16] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. Knowl. Data Eng.*, 18(1):92–106, 2006.

[17] K. Muralidhar and R. Sarathy. Security of random data perturbation methods. *ACM Trans. Database Syst.*, 24(4):487–493, 1999.

[18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology EUROCRYPT'99, LNCS 1592*, pages 223–238. Springer, 1999.

[19] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[20] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, New York, NY, USA, 2003. ACM Press.

[21] J. Vijayan. House committee chair wants info on cancelled dhs data-mining programs. *Computer World*, September 18, 2007.

[22] A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*. IEEE, 1982.

[23] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the twenty-seventh annual IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society, 1986.