IMPLEMENTATION OF CRYPTOGRAPHY FOR PRIVACY PRESERVING DATA MINING

Anand Sharma [1] and Vibha Ojha [2]

[1] CSE Deptt., MITS, Lakshmangarh, Sikar, Rajasthan
anand_glee@yahoo.co.in
[2] CSE Deptt., IITM, Gwalior, Madhya Pradesh
vibha.ojha@gmail.com

ABSTARCT

Privacy is one of the most important properties of an information system must satisfy, in which systems the need to share information among different, not trusted entities, the protection of sensible information has a relevant role. Thus privacy is becoming an increasingly important issue in many data mining applications. For that privacy secure distributed computation, which was done as part of a larger body of research in the theory of cryptography, has achieved remarkable results. These results were shown using generic constructions that can be applied to any function that has an efficient representation as a circuit. A relatively new trend shows that classical access control techniques are not sufficient to guarantee privacy when data mining techniques are used in a malicious way. Privacy preserving data mining algorithms have been recently introduced with the aim of preventing the discovery of sensible information. In this paper we will describe the implementation of cryptography in that data mining for privacy preserving.

KEYWORDS

Privacy preserving, Cryptography, Distributed Data Mining, Security.

1. INTRODUCTION

Privacy preserving data mining is an important property that any mining system must satisfy. So far, if we assumed that the information in each database found in mining can be freely shared. Consider a scenario in which two or more parties owning confidential databases wish to run a data mining algorithm on the union of their databases without revealing any unnecessary information. For example, consider separate medical institutions that wish to conduct a joint research while preserving the privacy of their patients. In this scenario it is required to protect privileged information, but it is also required to enable its use for research or for other purposes. In particular, although the parties realize that combining their data has some mutual benefit, none of them is willing to reveal its database to any other party.

The common definition of privacy in the cryptographic community limits the information that is leaked by the distributed computation to be the information that can be learned from the designated output of the computation. Although there are several variants of the definition of privacy, for the purpose of this discussion we use the definition that compares the result of the actual computation to that of an "ideal" computation: Consider first a party that is involved in the actual computation of a function (e.g. a data mining algorithm). Consider also an "ideal scenario", where in addition to the original parties there is also a "trusted party" who does not deviate from the behavior that we prescribe for him, and does not attempt to cheat. In the ideal scenario all parties send their inputs to the trusted party, who then computes the function and sends the appropriate results to the other parties. Loosely speaking, a protocol is secure if anything that an adversary can learn in the actual world it can also learn in the ideal world, namely from its own input and from the output it receives from the trusted party. In

essence, this means that the protocol that is run in order to compute the function does not leak any "unnecessary" information.

2. PRIVACY PRESERVING

Explosive progress in networking, storage and processor technologies has led to the creation of ultra large database that record unprecedented amount of transactional information. Privacy issues are further exacerbated now that the World Wide Web makes it easy for the new data to be automatically collected and added to databases. Privacy preserving protocols are designed in order to preserve privacy even in the presence of adversarial participants that attempt to gather information about the inputs of their peers. There are, however, different levels of adversarial behavior. Cryptographic research typically considers two types of adversaries: A semi-honest adversary (also known as a passive, or honest but curious adversary) is a party that correctly follows the protocol specification, yet attempts to learn additional information by analyzing the messages received during the protocol execution. On the other hand, a malicious adversary may arbitrarily deviate from the protocol specification. (For example, consider a step in the protocol where one of the parties is required to choose a random number and broadcast it. If the party is semi-honest then we can assume that this number is indeed random. On the other hand, if the party is malicious, then he might choose the number in a sophisticated way that enables him to gain additional information.) It is of course easier to design a solution that is secure against semi-honest adversaries, than it is to design a solution for malicious adversaries.

A common approach is therefore to first design a secure protocol for the semi-honest case, and then transform it into a protocol that is secure against malicious adversaries. This transformation can be done by requiring each party to use zero-knowledge proofs to prove that each step that it is taking follows the specification of the protocol. More efficient transformations are often required, since this generic approach might be rather inefficient and add considerable overhead to each step of the protocol. We remark that the semi-honest adversarial model is often a realistic one. This is because deviating from a specified program which may be buried in a complex application is a non-trivial task, and because a semi-honest adversarial behavior can model a scenario in which the parties that participate in the protocol are honest, but following the protocol execution an adversary may obtain a transcript of the protocol execution by breaking into a machine used by one of the participants.

3. PRIVACY PRESERVING COMPUTATION

In this section we will describe the various computation techniques which we are using for data.

3.1 Classification

Alice has a private database D1 and Bob has private database D2. How can Alice and Bob build a decision tree based on D1 D2 without disclosing the contents of their private database to each other? Several algorithms like ID3, Gain Ratio, Gini Index and many other can be used for Decision Tree.

3.2 Data Clustering

Alice has a private database D1 and Bob has private database D2. Alice and Bob want to jointly perform data clustering on D1 D2. This is primarily based on data clustering principle that tries to increase intra class similarity and minimize interclass similarity.

3.3 Mining Association Rules

Let Alice has a private database D1 and Bob has private database D2. If Alice and Bob wish to jointly find the association rules from D1 D2 without revealing the information from individual databases.

3.4 Data Generalization, Summarization and Characterization

Let Alice has a private database D1 and Bob has private database D2. If they wish to jointly perform data generalization, summarization or characterization on their combined database D1 □ D2, then this problem becomes an Secure Multiparty Communication problem.

3.5 Profile Matching

Alice has a database of hacker's profile. Bob has recently traced a behavior of a person, whom he suspects a hacker. Now, if Bob wants to check whether his doubt is correct, he needs to check Alice's database. Alice's database needs to be protected because it contains hacker's related sensitive information. Therefore, when Bob enters the hacker's behavior and searches the Alice's database, he can't view his whole database, but instead, only gets the comparison results of the matching behavior.

3.6 Fraud Detection

Two major financial organizations want to cooperate in preventing fraudulent intrusions into their computing system, without sharing their data patterns, since their individual private database contains sensitive data.

4. SECURE COMPUTATION AND PRIVACY PRESERVING DATA MINING

There are two distinct problems that arise in the setting of privacy-preserving data mining. The first is to decide which functions can be safely computed, where safety means that the privacy of individuals is preserved. For example, is it safe to compute a decision tree on confidential data in an organization and publicize the resulting tree? For the most part, we will assume that the result of the data mining algorithm is either safe or deemed essential. Thus, the question becomes how to compute the results while minimizing the damage to privacy. For example, it is always possible to pool all of the data in one place and run the data mining algorithm on the pooled data. However, this is exactly what we don't want to. Thus, the question we address is how to compute the results without pooling the data, and in a way that reveals nothing but the final results of the data mining computation. This question of privacy-preserving data mining is actually a special case of a long-studied problem in cryptography called secure multiparty computation. This problem deals with a setting where a set of parties with private inputs wish to jointly compute some function of their inputs. Loosely speaking, this joint computation should have the property that the parties learn the correct output and nothing else, even if some of the parties maliciously collude to obtain more information. Clearly, a protocol that provides this guarantee can be used to solve privacy-preserving data mining problems of the type discussed above.

5. CRYPTOGRAPHY: OBLIVIOUS TRANSFER

We describe here results of a body of cryptographic research that shows how separate parties can jointly compute any function of their inputs, without revealing any other information. As we argued above, these results achieve maximal privacy that hides all information except for the designated output of the function. This body of research attempts to model the world in a way which is both realistic and general. While there are some aspects of the "real world" that are not modeled by this research, the privacy guarantees and the generality of the results are quite remarkable.

Oblivious transfer is a basic protocol that is the main building block of secure computation. It might seem strange at first, but its role in secure computation should become clear later. (In fact, it was shown by Kilian [11] that oblivious transfer is sufficient for secure computation in the sense that given an implementation of oblivious transfer, and no other cryptographic primitive, one could construct any secure computation protocol.)

Oblivious transfer is often the most computationally intensive operation of secure protocols, and is repeated many times. Each invocation of oblivious transfer typically requires a constant number of invocations of trapdoor permutations (i.e. public-key operations, or exponentiations). It is possible to reduce the amortized overhead of oblivious transfer to one exponentiations per a logarithmic number of oblivious transfers, even for the case of malicious adversaries [15].

The problem of "oblivious polynomial evaluation" (OPE) involves a sender and a receiver. The sender's input is a polynomial Q of degree k over some finite field f and the receiver's input is an element z ε f (the degree k of Q is public). The protocol is such that the receiver obtains Q (z) without learning anything else about the polynomial Q, and the sender learns nothing. That is, the problem considered is the private computation of the function (Q, z) $\rightarrow$ (λ,Q (z)). This problem was introduced in [14], where an efficient solution was also presented. The overhead of that protocol is O (k) exponentiations (using methods suggested in [15]). (Note that this protocol maintains privacy in the face of a malicious adversary. In the semi-honest case a simpler OPE protocol can be designed based on any homomorphic encryption scheme, with an overhead of O (k) computation and O(k | f |) communication.)

The main motivation for using OPE is to utilize the fact that the output of a k degree polynomial is (k + 1)-wise independent. Another motivation is that polynomials can be used for approximating functions that are defined over the Real numbers.

6. THE TWO-PARTY CASE

Yao's two-party protocol is pretty efficient, as long as the size of the inputs, and the size of the circuit computing the function, are reasonable. In fact, for many functions the efficiency of Yao's generic protocol is comparable to that of protocols that are targeted for computing the specific function. We describe here a distributed scenario of computing the ID3 algorithm, where Yao's protocol is obviously too costly. On the other hand, a specialized protocol can be designed for computing this algorithm, which uses Yao's protocol as a primitive.

We are interested in a scenario involving two parties, each one of them holding a database of different transactions, where all the transactions have the same set of attributes (this scenario is also denoted as a "horizontally partitioned" database). The parties wish to compute a decision tree by applying the ID3 algorithm to the union of their databases.

A naive approach for implementing a privacy preserving solution is to apply the generic Yao protocol to the ID3 algorithm. This approach encounters two major obstacles. First, the size of the databases is typically very large. As each transaction can have many attributes, and there might be millions of transactions, the encoding of each party's input might require hundreds of millions of bits. This means that the computational overhead of running an oblivious transfer per input bit might be very high.

Most cryptographic protocols, however, compute functions over finite fields. Even if the circuit computes an approximation to the logarithm, this computation involves evaluating polynomials and therefore requires computing multiplications and exponentiations. An additional problem is that running ID3 involves many rounds. The part of the circuit computing the i^{th} round depends on the results of the previous i−1 rounds. A naïve implementation could require an encoding of many copies of this step, each one of them corresponding to a specific result of the previous rounds.

A key observation is that each node of the tree can be computed separately, with the output made public, before continuing to the next node. In general, private protocols have the property that intermediate values remain hidden. However, in the case of ID3 some of these intermediate values (specifically, the assignments of attributes to nodes) are actually part of the output and may therefore be revealed. Once the attribute of a given node has been found, both parties can separately partition their remaining transactions accordingly for the coming recursive calls. This means that private distributed ID3 can be reduced to privately finding the attribute with the highest information gain. (This is a slightly simplified argument as the other steps of ID3 must also be carefully dealt with. However, the main issues arise within this step.)

The overhead of the protocol described above involves:

- Alice and Bob engaging in an oblivious transfer protocol for every input wire of the circuit that is associated with Bob's input,
- Alice sending Bob tables of size linear in the size of the circuit,
- Bob decrypting a constant number of cipher texts for every gate of the circuit (this is the cost incurred in evaluating the gates).

The computation overhead is dominated by the oblivious transfer stage, since the evaluation of the gates uses symmetric encryption which is very efficient compared to oblivious transfers that require modular exponentiations (this holds for small circuits; if the circuit is large then the circuit computation may begin to dominate). The computation overhead is therefore roughly linear in the length of Bob's input. The number of rounds of the protocol is constant. (namely, the variant described here has two rounds using the two-round oblivious transfer protocols of [5, 6, 15]).

The communication overhead is linear in the size of the circuit. (The variant of the protocol described in [22], which provides security against malicious adversaries, requires sending s copies of the circuit in order to limit the probability of cheating to be exponentially small in s. See also [17] for a different variant, which provides security against malicious adversaries at the cost of applying public key operations for every gate.)

A major factor dominating the overhead is, therefore, the size of the circuit representation of *f*. There are many functions for which we do not know how to create linear size circuits (e.g. functions computing multiplications or exponentiations, or functions that use indirect addressing). However, there are many other functions, notably those involving additions and comparisons, which can be computed by linear size circuits. The size of the input should also be reasonable. For example, we cannot expect that two parties, each of them holding a database with millions of entries, could run the protocol for computing a function whose inputs are the entire databases.

7. THE MULTI-PARTY CASE

The multi-party case involves three or more parties that wish to compute some function of their inputs without leaking any unnecessary information. In the multi-party scenario, there are protocols that enable the parties to compute any joint function of their inputs without revealing any other information about the inputs. That is, compute the function while attaining the same privacy as in the ideal model. This was shown to be possible in principle by Goldreich, Micali and Wigderson [10], Ben-Or, Goldwasser and Wigderson [3], and by Chaum, Crepau and Damgard [4], for different scenarios. These constructions, too, are based on representing the computed function as a circuit and evaluating it. The constructions do have, however, some additional drawbacks, compared to the two-party case:

- The computation and communication overhead of the protocol is linear in the size of the circuit, and the number of communication rounds depends on the depth of the circuit1, unlike the two-party case where the number of rounds is constant. Furthermore, the protocol that is run for every gate of the circuit is more complex than the computation of a gate in the two-party case, especially in the malicious party scenario, and requires public-key operations (although the overhead is still polynomial).
- The multi-party protocols require each pair of parties to exchange messages (in order to compute each gate of the circuit). The required communication graph is, therefore, a complete graph, whereas a sparse communication graph could have been sufficient if no security was required. In many applications, for example applications run between a web server and many clients, it is impossible to require all pairs of parties to communicate.
- The security of the multi-party protocols is assured as long as there is no corrupt coalition of more than one half or one third of the parties (depending on the scenario). In many situations, however, it is impossible to ensure that the number of corrupt

parties is smaller than such a threshold (for example, consider a web application in which anyone can register and participate, and which, therefore, enables an adversary to register any number of corrupt participants). In such cases the security of the protocol is not guaranteed.

Compared to the two-party case, however, it is harder to apply the generic constructions to actual scenarios. To illustrate this point we consider the case of running a secure computation for computing the result of an auction, where there is an obvious motivation for privacy and security, and also certain restrictions on the operation of the parties. The auction application, discussed in [16], is not related to data mining, but it does exemplify some of the difficulties of the multiparty case. The discussion below applies for any function that can be computed by a circuit of reasonable size.

The auction scenario is that of a "sealed bid" auction, and consists of an auctioneer and many bidders. Each bidder submits a single secret bid (i.e. the bid is sealed in an envelope). There is a known decision rule, whose inputs are the submitted bids, and whose output is the identity of the winning bidder and the amount that this bidder has to pay. For example, in an "English auction" the winning bidder is the bidder who offered the highest bid, and he has to pay the amount of his bid. In the second-price, or Vickrey, type of auction (which has some nice properties that are outside the scope of this paper) the winner is the highest bidder and he has to pay the amount of the second highest bid. Bidding is allowed until some point in time, and at that stage the decision rule is applied to the submitted bids.

In the physical world bids are submitted in sealed envelopes that are kept secure until the end of the bidding period, and are then opened by the auctioneer. In the virtual world we would like to keep the bids secret during the bidding period, but we could also attempt to hide all information afterwards, except for the identity of the winning party and the amount he has to pay. For example, in the case of a Vickrey auction the auctioneer's output could be limited to the identity of the highest bidder (but not the value of his bid), and the value of the second highest bid (but not the identity of the second highest bidder). This is more privacy than can be achieved in the physical world. (In fact, some of the suggested explanations for the unpopularity of second price auctions are based on possible attacks that a malicious auctioneer can mount if he learns the bid value of the highest bidder. This phenomenon is inevitable in the real world, but can be avoided if a privacy preserving protocol is used to compute the result of the auction.)

Privacy preserving multi-party computation can be reduced to the two-party case. Namely, it is possible to use the generic two-party protocol to compute a function in the multi-party scenario. Such a reduction is described in [16]. Before describing the highlights of the reduction we first describe the advantages of this approach.

7.1 Trust

In order to use the two-party construction it is assumed that there are two special parties, and privacy is preserved as long as these two parties do not collude. Namely, a collusion of any number of parties (even a majority of the parties) that does not include both special parties does not affect the privacy and security of the protocol. Protocols with this security assurance might seem weaker than protocols that are secure against collusions of say, any coalition of less than one half of the parties. After all, there is a coalition of just two parties – the two special parties, is able to break the security of the system. Consider however a scenario where most of the parties are users (e.g. bidders) that have not established trust relationships between themselves, and there are one or more central parties that are more established. For example, in the auction scenario we can assume that the two special parties are the auctioneer and another party which we denote as the "issuer", and which can be, for example, an accounting firm. We know that an adversary can register many fake bidders in order to control a majority of the participating parties. It seems harder, though, for the adversary to be able to control insiders of both special parties, i.e. in the auctioneer's organization and in the accounting firm.

7.2 Independence of Inputs

Corrupted parties must choose their inputs independently of the honest parties' inputs. This property is crucial in a sealed auction, where bids are kept secret and parties must fix their bids independently of others. We note that independence of inputs is not implied by privacy. For example, it may be possible to generate a higher bid, without knowing the value of the original one. Such an attack can actually be carried out on some encryption schemes.

7.3 Communication

We can design the reduction such that each of the "simple" participating parties should only communicate with one of the special parties (e.g. the auctioneer), and should only send a single message to this party. This property greatly simplifies the required communication infrastructure, and enables to run the protocol without requiring all parties to be online at the same time (in fact, compared to a protocol that provides no security at all, the only new communication channel that is introduced by the secure protocol is the channel between the two special parties). When all the "simple" parties finish sending their messages, the two special parties run a short protocol to complete the computation of the function.

7.4 Privacy

No party should learn anything more than its prescribed output. In particular, the only information that should be learned about other parties' inputs is what can be derived from the output itself. For example, in an auction where the only bid revealed is that of the highest bidder, it is clearly possible to derive that all other bids were lower than the winning bid. However, this should be the only information revealed about the losing bids.

7.5 Correctness

Each party is guaranteed that the output that it receives is correct. To continue with the example of an auction, this implies that the party with the highest bid is guaranteed to win, and no party including the auctioneer can alter this.

7.6 Efficiency

The protocol evaluates a circuit representation of the function. The overhead per gate and per input bit is as in the two-party construction, and is lower than in the multi-party constructions.

7.7 Guaranteed Output Delivery

Corrupted parties should not be able to prevent honest parties from receiving their output. In other words, the adversary should not be able to disrupt the computation by carrying out a "denial of service" attack.

7.8 Fairness

Corrupted parties should receive their outputs if and only if the honest parties also receive their outputs. The scenario where a corrupted party obtains output and an honest party does not should not be allowed to occur. This property can be crucial, for example, in the case of contract signing. Specifically, it would be very problematic if the corrupted party received the signed contract and the honest party did not.

The protocol is run with the two special parties taking the roles of the two parties in the two-party case. The issuer prepares a circuit for computing the function. This circuit might have many inputs of different parties – for example, the inputs might be the bids of the different bidders. The issuer encodes the circuit as in the two-party case, by choosing garbled values for the wires and preparing tables for every gate. The other special party (the auctioneer) is

responsible for computing the result of the circuit. In order to do that it should receive the tables that were prepared by the issuer, and one garbled value for every input wire, namely the value that corresponds to the input bit associated with that wire. Once it receives the garbled values of all input wires it can compute the output of the circuit.

Given the proxy oblivious transfer protocol, the rest of the implementation is simple. Each bidder engages in a proxy oblivious transfer for each of its input bits. The input of the bidder to this protocol is the value of the input bit. The sender is the issuer, and its two inputs are the two garbled values that are associated with the corresponding input wire. The receiver is the auctioneer, and it learns the garbled value that corresponds to the input bit. This protocol consists of a single message that is sent from the bidder to the auctioneer, and then a round of communication between the auctioneer and the issuer. The auctioneer can actually wait until it receives messages from all the bidders before it runs the round of communication with the issuer in parallel for all input bits. The main computational overhead of the protocol is incurred by the proxy oblivious transfers, and is the same as in the two-party case – a proxy oblivious transfer must be executed for every input wire. Estimates in [16] show that this method can be used to securely implement Vickrey auctions that involve hundreds of bidders.

8. CONCLUSIONS

Cryptographic protocols for secure computation achieved remarkable results: it was shown that generic constructions can be used to compute any function securely and it was also demonstrated that some functions can be computed even more efficiently using specialized constructions. Still, a secure protocol for computing a certain function will always be more costly than a naive protocol that does not provide any security. By making use of cryptographic techniques to store sensitive data and providing access to the stored data based on an individual's role, we ensure that the data is safe from privacy breaches. This paper was intended to demonstrate basic ideas from a large body of cryptographic research on secure distributed computation, and their applications to data mining. We described in brief the definitions of security, and the generic constructions for the two-party and multi-party scenarios. We showed that it is easier to design an implementation based on the constructions for the two-party case than it is to design one based on the multi-party constructions. The main parameter that affects the feasibility of implementing a secure protocol based on the generic constructions is the size of the best combinatorial circuit that computes the function that is evaluated. We believe that further research in this area is crucial for the development of secure and efficient protocols in this field.

REFERENCES

[1] D. Beaver, S. Micali and P. Rogaway, The round complexity of secure protocols, Proc. of 22nd ACM Symposium on Theory of Computing (STOC), pp. 503-513, 1990.

[2] M. Bellare and S. Micali, Non-Interactive Oblivious Transfer and Applications, Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, Vol. 435, Springer-Verlag, 1997, pp. 547-557.

[3] M. Ben-Or, S. Goldwasser and A. Wigderson, Completeness theorems for non cryptographic fault tolerant distributed computation, Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC), ACM, 1988, pp. 1–9.

[4] D. Chaum, C. Crepeau and I. Damgard, Multiparty unconditionally secure protocols, Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC), ACM, 1988, pp. 11–19.

[5] S. Even, O. Goldreich and A. Lempel. A Randomized Protocol for Signing Contracts. Communications of the ACM, 28(6):637-647, 1985.

[6] O. Goldreich. Foundations of Cryptography: Volume 2 { Basic Applications. Cambridge University Press, 2004

[7] Jaideep Vaidya and Chris Clifton, "Leveraging the 'multi' in Secure Multiparty Computation," WPES'03 October 30, 2003, Washington, DC, USA, ACM Transaction 2003, pp120-128.

[8] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, Michael Y. Zhu, "Tools for Privacy Preserving Data Mining". international conference on knowledge discovery and data mining, Vol. 4, No. 2, 2002, pp. 1-8.

[9] Anand Sharma and vibha ojha ""Privacy preserving Data Mining by Cryptography" in Springer-LNCS-CICS- Vol:89, "Recent Trends in Network Security and Applications" .pp.576-581.2010.

[10] O. Goldreich, S. Micali and A. Wigderson, How to Play any Mental Game - A Completeness Theorem for Protocols with Honest Majority, Proceedings of the 19th Annual Symposium on the Theory of Computing (STOC), ACM, 1987, pp. 218–229.

[11] J. Kilian, Founding cryptography on oblivious transfer, ACM STOC '88, pp. 20-31.

[12] Y. Lindell and B. Pinkas, Privacy Preserving Data Mining, Journal of Cryptology, Vol. 15, No. 3, pp. 177-206, 2002.

[13] M. Luby, Pseudorandomness and Cryptographic Applications, Princeton Computer Science Notes, 1996.

[14] M. Naor and B. Pinkas, Oblivious Transfer and Polynomial Evaluation, Proceedings of the 31th Annual Symposium on the Theory of Computing (STOC), ACM, 1999, pp. 245–254.

[15] M. Naor and B. Pinkas, Efficient Oblivious Transfer Protocols, Proceedings of 12th SIAM Symposium on Discrete Algorithms (SODA), January 7-9 2001, Washington DC, pp. 448–457.

[16] M. Naor, B. Pinkas and R. Sumner, Privacy Preserving Auctions and Mechanism Design, Proc. of the 1st ACM conference on Electronic Commerce, November 1999.

[17] S. Jarecki and V. Shmatikov. Eficient Two-Party Secure Computation on Committed Inputs. In EUROCRYPT 2007, Springer-Verlag (LNCS 4515), pages 97-114, 2007.

[18] Rebecca Wright, "Progress on the PORTIA Project in Privacy Preserving Data Mining," A data surveillance and privacy protection workshop held on 3rd June 2008.

[19] M. O. Rabin, How to exchange secrets by oblivious transfer, Technical Memo TR-81, Aiken Computation Laboratory, 1981.

[20] J.E. Savage, Computational work and time on finite machines, Journal of the ACM, 19(4), pp. 660-674, 1972.

[21] A. C. Yao, How to generate and exchange secrets, Proceedings 27th Symposium on Foundations of Computer Science (FOCS), IEEE, 1986, pp. 162–167.

[22] Y. Lindell and B. Pinkas. An Eficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries In EUROCRYPT 2007, Springer-Verlag (LNCS 4515), pages 52-78, 2007.

AUTHORS

Anand Sharma

He received his Masters degree from ABV-IIITM, Gwalior, in 2008. Currently, he is an Assistant Professor at MITS, Lakshmangarh. His interests are Data Mining, Network & Information Security and Quantum Cryptography. He is having around 20 publications in International / National Journals and conferences.

Vibha Ojha

She has done her Bachelor of Engineering Degree in Computer Science & Engineering form IITM, Gwalior. Her research areas are Quantum Cryptography, Data mining and Network Security. She is having around 13 publications in International / National Journals and conferences.