Privacy-Preserving Minimum Spanning Tree Algorithms in Static Semi-honest Model

Koteswara Rao Ch* and Kunwar Singh
National Institute of Technology, Tiruchirappalli, Tamil Nadu, 620015, India
{406116006, kunwar}@nitt.edu

Abstract

In 1982, Andrew Yao introduced secure two-party computation (2PC) for the so-called millionaire's problem. The problem is about two millionaires Alice and Bob, interested to determine who is wealthier without revealing their actual private property values. Goldreich generalized the secure two-party computation and formalized the secure multi-party computation. Suppose two telephone companies wish to merge to provide better services to end users. Each company has a cost function for connecting any pair of houses. They want to connect every house with minimum cost in merged company. Mathematically, given two graphs G_1,G_2 they want to compute MST(min(G_1,G_2)). Before merging both companies they want to know whether merging will benefit them or not without reveling cost function for any pair of houses. Based on the secure multi-party computation paradigm, we propose new algorithms for privacy-preserving computation of minimum spanning tree. Our protocols offers security against semi-honest adversaries.

Keywords: Secure multiparty computation, privacy, minimum spanning tree

1 Introduction

In 1982, Andrew Yao [13] introduced secure two-party computation (2PC) for the so-called millionaires' problem . The millionaires' problem is about two millionaires Alice and Bob, interested to determine who is wealthier without revealing their actual private property values. Yao has presented a constant round protocol in a stable manner for computing a set of two-party functionality in semi-honest model adversaries. Goldreich *et al.* [7] generalized the secure two-party computation and formalized the secure multi-party computation. The mathematical model for secure multi-party computation formally defined as : parties $P_1, P_2, ...P_n$ want to jointly compute the function y = f($x_1, x_2, ..., x_n$) without revealing the secret value x_i to each other.

In Asiacrypt 2005, Justin Brickell *et al.* [3] presented privacy-preserving all pairs shortest path algorithm and single source shortest path algorithm in semi-honest model. They investigated the problem by taking two internet providers (mutually distrustful parties) each having their private networks and wish to merge with each other to compute efficient joint graph network. Abdelrahaman Aly *et al.* [1] presented shortest path and maximum flow problems with the help of Arithmetic Black-Box functionality in secure multi-party computation environment. Tomas Toft [12] presented secure data structures based on multiparty computation with the Arithmetic Black-Box functionality.

Suppose two telephone companies wish to merge to provide better services to end users. Each company has a cost function for connecting any pair of houses. They want to connect every house with minimum cost in merged company. Mathematically, given two graphs G_1,G_2 they want to compute MST(min(G_1,G_2)). Before merging both companies want to know whether merging will benefit them or not without reveling cost function for any pair of houses.

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 5, Article No. 1 (March 15, 2019)

*Corresponding author: Research Scholar, Department of Computer Science and Engineering, NIT, Tirchy, Tamil Nadu, 620015, INDIA, Tel: +91-9959946303

Contribution: Working with secure multi-party computation paradigm, we present new algorithms of minimum spanning tree for combined graph (G_1,G_2) with out revealing cost functions of the graphs to each other. The algorithms are privacy-preserving version of minimum spanning tree algorithms (Kruskal's and Prim's) in semi-honest model. The technique is belonging to the family of the cryptographic idea of secure multi-party computation in a privacy-preserving manner. We used Yao's [14, 8] protocol as a sub-protocol to compute minimum in our scheme.

2 Preliminaries

This section describes basic definitions that are required to understand our scheme.

Negligible: We may conclude that a negligible function denoted as negl(n) in n, if it is small value than the inverse of any other polynomial function in n for large n. Mathematically, function f(n) is negligible function if f(n) = o(n^c) for all the constant c and n $\geq n_0$, where o is standard little-oh notation.

Computational indinstinguishability: Let $\{X_p\}_p$ and $\{Y_p\}_p$ be ensembles where X_p's and Y_p's are probability distributions over $\{0,1\}^{l(s)}$ for some polynomial value $l(s)$. We can say that $\{X_p\}_p$ and $\{Y_p\}_p$ are computationally indistinguishable if for all the non-uniform probabilistic polynomial time $\{D_p\}_{p\in N}$, there must exist a negligible function ε such that for all $p \in N$ such that

$$|Pr[k \leftarrow X_p : D(k) = 1] - Pr[k \leftarrow Y_p : D(k) = 1]| \leq \varepsilon(p).$$

In this particular case we will write $X \stackrel{c}{\equiv} Y$.

2.1 Secure two-party computation

Andrew Yao in 1982, introduced the concept of secure two-party computation (2PC) for the so-called millionaires' problem. The main goal of two-party computation is creating a generic protocol that have two parties to combine and calculate an arbitrary function on their related private input values without leaking the secret values to each other. The best well known examples of two-party computation is Yao's millionaire problem, in this two parties, Alice and Bob, are millionaires wishing to determine who is richer without revealing their actual wealth details.

As a stepping stone of our constructed algorithms, we deployed privacy preserving protocols for calculation of minimum that is min(p,q). In this we are using the Yao's protocol as a sub protocol which is used for calculating private minimum of P_1 and P_2, that is m = min (m_1, m_2). The definitions for different settings of secure two-party computation are found in [2, 4, 6].

2.2 Secure multi-party computation

Consider the setting, where a collection of n number of parties $P_1, ..., P_n$ wish to compute a joint function f on their n respective private input values and receive the resulting output values. The problem statement of secure computation is to guarantee the security of this function evaluation in the existence of semi-honest adversary model. The security guarantees here would be to ensure correctness of the output and secrecy of the data held by the parties.

Let us formalize what happens in the real world. All parties interact with each other according to the protocol specification in the presence of a real-life adversary model who controls a certain set of n number of parties. At the end of the computation, the parties which are not corrupted, output no matter what is specified in the protocol. The adversary, controlling the corrupted parties, outputs some arbitrary

random value. This value may be including information about that the adversary gathered during the computation.

Let f be a randomized functionality from n number of inputs to n number of outputs (i.e.) $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$. A protocol λ computes f if, when parties $P_1,...,P_n$ run λ on inputs $k, x_1,...,x_n$, they output $y_1,...,y_n$ that is distributed according to the output function $f(x_1,...x_n)$. The definitions for different settings of secure multi-party computation are found in [2, 4, 6]. With the help of oblivious transfer [10, 11] and secure function evaluation [9] it can be implemented easily.

2.2.1 Central properties of MPC

- *Privacy:* Only the prescribed output will be revealed. No other party should learn nothing more than its specified out-put. In general, if n number of parties are interacting with each other to compute an arbitrary function, the other opponent parties know about the final output of the service rather than private values of the others.

- *Correctness:* This property says that each party was receiving correct output. That means all the parties that are involved in a secure function evaluation they are guaranteed to get the correct output.

- *Independence of Inputs:* In this, the inputs of the corrupted parties should be chosen independently from the honest party inputs. Let us take a simple example case of auctions. This property is critical in a sealed secure auction when bids are kept secret and secure, and parties fix their inputs independently of the others.

- *Guaranteed Output Delivery:* In this property even though the corrupted parties applying denial of service attack, it must produce guaranteed output. The output is delivered to honest parties even in the presence of corrupted parties. We say that adversary could not be able to prevent vague parties from receiving their respective output.

- *Fairness:* This says that the corrupted party can receive the output if and only if the original party receives the respective output. The meaning of this is if one party receives the output then all the parties will receive. Let us consider a corrupted party gains output as contract signing, and the original party does not receive, this is not the fair case, it is a crucial property. Both the guaranteed output delivery and fairness are almost similar.

2.2.2 Adversarial behavior in MPC

- *Semi-honest adversaries:* The adversary can study the internal information of the corrupted parties. These are also called as passive adversarial models. In this dishonest parties also exactly follows the specification of the protocol. While executing transcript messages in the protocol, there is a chance of leaking additional information by attempts to learn. It is a weak adversary model. These are also named as ”honest-but-curious.”

- *Malicious adversaries:*

  The adversary can take complete control over the parties and can make them (mis)behave in any desired manner. In the malicious model, restrictions are not placed on any of the participants. Thus parties are completely free to spoil the actions that are placed in the protocol. These are ”active” adversaries. These are very stronger than the semi-honest adversaries model. These adversaries will deviate from the protocol execution. It is difficult to provide security when malicious adversaries exist, however with strong protocol scheme we can overcome these type of adversaries. Let

us consider the interaction between different intelligence agencies; for the sake of security, they wouldn't give free accessing to the confidential data.

2.2.3 Applications/Examples of MPC

- *Secure Auction:* Every party having a bid amount as a private input data. Goal of this action is to compute maximum bid among several bid's and revealing the final winner bid amount only nothing other than that.
- *Satellite Collision:* In 1994 two satellites collided resulting a big loss to two countries. The orbit information of a satellite is private information owned by a country. To prevent collision from different satellites in the orbit, the goal is to make the collision probability to zero without leaking exact accurate information of satellites.
- *Privacy Preserving Data Mining:* Hospitals have patient information which is private data. Many hospitals wants to know the exact count of patients who are suffering from a particular disease without revealing patient information.

2.3 Security model for two party protocol for semi-honest adversaries:

The security model for two party setting protocol in the existence of semi-honest adversaries is adapted from Yao [14] and Goldreich *et al.* [6]. The definition is formalized according to simulator paradigm. If a protocol is simulated based on input and output of the parties those who are participating in the protocol, then the parties in the protocol learns nothing. Intuitively a protocol can be secure, if the parties view in a protocol can be simulatable given only that party's input and output.

2.3.1 Notations:

We used the list of notations that are prescribed below for the definition of privacy.

- Let us take probabilistic polynomial time function $F = (F_1, F_2)$ and let λ is a two-party protocol that computes F.
- The *view* of party *i* while executing λ on (p,q) is denoted as $view_i^{\lambda}(p,q)$, where i $\in \{1,2\}$.
- The output of party *i* while executing λ on (p,q) is denoted as $output_i^{\lambda}(p,q)$, it can be calculated from its original own view of the execution.

Protocol λ computes securely the deterministic functionality f in the existence of static semi-honest adversaries if there exist non uniform probabilistic polynomial time simulators S_1 and S_2 such that

$$\{S_1(p, f(p,q))\}_{p,q\in\{0,1\}^*} \equiv \{view_1^{\lambda}(p,q)\}_{p,q\in\{0,1\}^*}$$
$$\{S_2(q, f(p,q))\}_{p,q\in\{0,1\}^*} \equiv \{view_2^{\lambda}(p,q)\}_{p,q\in\{0,1\}^*}$$

here $|p| = |q|$.

3 Our Privacy-Preserving Minimum Spanning Tree Algorithms on Joint Graphs

Now we present the construction of privacy preserving minimum spanning tree algorithms in semi honest adversary model. Let us assume there are two private parties named as G_1 and G_2 with strictly positive

edge costs, let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two complete connected graphs with equal number of vertices and equal number of edges but with different edge costs. The incomplete graphs are allowed with ∞ assigning to their edge cost.

Suppose two different telephone operated cell phone companies wishing to merge with each other, each company is having private network in terms of cost function value connecting any number of pair of houses in the city. They want to connect every individual house with least cost in merged company without revealing their original cost value of the function for connecting the pair of vertices (houses). The solution to this problem is constructing Minimum Spanning Tree from the joint graph. Our first proposed algorithm is built on the concept of Kruskal's and the second proposed algorithm is built on the Prim's concept.

3.1 Privacy Preserving Minimum Spanning Tree using Kruskal's (PPMSTK)

The minimum spanning tree is a classical graph theory problem that is used to find minimum spanning tree in between all the vertices that are present in a connected weighted graph [5]. Spanning tree is defined as sub graph $G' = (V', E', w')$ of the given connected weighted main graph G=(V, E, w) which covers all the vertices with n-1 edges without forming cycles. Minimum Spanning Tree is defined as same as spanning tree with minimum edge costs. If a graph is given with strictly positive weighted edges, the minimum spanning tree is set of edges with the minimum weight which satisfies tree properties *connected*, *acyclic* and *consists of |v|-1 edges*.

The basic idea of our problem is let $G_1 = (V_1, E_1)$ a graph of first party and $G_2 = (V_2, E_2)$ a graph of the second party be the graphs corresponding to two different telephone companies. Our algorithm computes minimum spanning tree from graph G_1 and graph G_2 without revealing weights of graph to each other.

Our Scheme:

1. We used a variable k for the convenience of the notation, it is set to 0, it describes iteration count.
2. Both the parties in their respective private graphs, privately compute the minimum length
$$m_1^{(k)} = \text{minimum}\{w_1(e_{G_1})\} \quad (1)$$
$$m_2^{(k)} = \text{minimum}\{w_2(e_{G_2})\} \quad (2)$$
3. Using Yao's privacy-preserving generic minimum protocol, compute the smallest edge. This does not reveal about other values
$$m^{(k)} = \min\{m_1^{(k)}, m_2^{(k)}\} \quad (3)$$
4. Create a public graph and mark it with the min edge in step 3 to find MST T, initially T is empty
5. If cycle does not form, add min edge to T i.e T = $\{T \cup m\}$, increment k value and go to step 2 to find next minimum edges of the private graphs.
Else discard the selected minimum edge m.
6. Repeat this process until k $<$ n-1.

The final graph is MST using kruskal's in privacy preserving manner.

Correctness: To prove this algorithm is correct, it is already known that kruskal's algorithm is correct, we used the same concept here therefore our scheme is also correct.

Our algorithm preserves privacy in semi-honest adversary model.

Proof (Privacy). We have to prove that real transcript of our private minimum protocol and simulated transcript of a private minimum protocol are computationally indistinguishable. As it is already proved that Yao's two-party protocol preserves privacy i.e real transcript of the two-party protocol of Yao and simulated transcript of the two-party protocol of Yao are computationally indistinguishable.

Here we will show that if our protocol does not preserve privacy in semi-honest adversary model then Yao's two-party protocol also does not preserve privacy in semi-honest adversary model. Since we know that Yao's two-party protocol preserves privacy, so our protocol also preserves privacy.

Let us describe a simulator for P_1, the simulator is given P_1's input to the protocol, the output of the protocol is f(p,q) = G′, for party P_1 and P_2 using the same simulator. But two parties are not similar if the minimum sub protocol exists in between P_1 and P_2. The parties P_1 and P_2 are having identical statements. Let us assume there are n number of sub protocol rounds. The party P_1's view is written as

$$\{RT^{min}(p_1,q_1), RT^{min}(p_2,q_2),...,RT^{min}(p_i,q_i),...,RT^{min}(p_n,q_n)\} \quad (4)$$

Here RT^{min} describes real (original) transcript of the minimum protocol.
The simulator such as polynomial function of G′ can be used to compute each of the output of executions of the protocol, which will be denoted as $h_i^{min}(G')$.
The computed P_1's input denoted as $g_i^{min}(p,G')$, it also computed by a polynomial function of x and G′ at each of the protocol execution. Using the minimum sub protocol simulators to generate the simulated transcripts.

$$\{ST^{min}(g_1^{min}(p,G'),h_1^{min}(G')),...,ST^{min}(g_n^{min}(p,G'),h_n^{min}(G'))\} \quad (5)$$

Where ST^{min} describes the Simulated Transcript of private minimum sub protocol in semi-honest model. By using views of simulated transcripts for the minimum sub protocol we can prove the hybrid arguments. We can define distribution of hybrid H_i in such a way that the first i minimum sub protocols were simulated and the last $n-i$ transcripts are real one. Let the value H_i(x,y) demonstrates the distribution

$$\{ST^{min}(g_1^{min}(p,G'),h_1^{min}(G')),...,RT^{min}(p_{i+1},q_{i+1}),...,RT^{min}(p_n,q_n)\} \quad (6)$$

We now have to prove H_0(p,q) $\equiv$ H_n(p,q) by demonstrating for all the i values, $H_i(p,q) \equiv H_{i+1}(p,q)$.

By contradiction, let us take that there is a D, denotes PPT distinguisher and pol(.) a polynomial, such that infinite number of many n′s (p,q $\in \{0,1\}^n$)

$$|Pr[D(H_0(p,q)) = 1] - Pr[D(H_n(p,q)) = 1]| > \frac{1}{pol(n)}$$

It is saying that for infinitely many number of p and q, there is an i.

$$|Pr[D(H_i(p,q)) = 1] - Pr[D(H_{i+1}(p,q)) = 1]| > \frac{1}{n\ pol(n)}$$

To contradict the security mechanism of the minimum sub protocol let us use D. That the solitary dissimilar between H_i(p,q) and H_{i+1}(p,q) is of the $(i+1)^{th}$ transcript of minimum according to $view^{\lambda}(p_{i+1},q_{i+1})$ in H_i and accordingly to $S^{\lambda}(p_{i+1},q_{i+1})$ in H_{i+1}. Furthermore, given p,q,i and a view v it is possible to construct a distribution H such that if v is taken from view_1^{λ} then H = H_i(p,q) and if v is taken from S_1^{λ} then H = H_{i+1}(p,q). It follows the infinite number of many inputs, however it is possible to distinguish the view of P_1 in real by finding minimum simulated view with the same probability. That it is possible to identify H_i(p,q) from H_{i+1}(p,q). However, there is a contradiction in the security of minimum

sub-protocol. Therefore we can come to an end that $H_0(p,q) \stackrel{c}{\equiv} H_n(p,q)$. We have shown that for each execution of minimum sub protocol where input is P_1's sub protocol and output is P_1's input function and entire MST protocol.

3.2 Privacy Preserving Minimum Spanning Tree using Prim's (PPMSTP)

Generic Prim's algorithm is used to find find minimum spanning tree in a given connected, weighted single graph [5]. While using this technique it leaks supplementary additional information regarding other edges which are adjacent to a particular vertex.

Let us take $G_1 = (V_1, E_1)$ be the graph of first party and $G_2 = (V_2, E_2)$ be the graph of second party as two private graphs, which are having same set of vertices and same set of edges with different strictly positive edge costs. Choose an arbitrary starting vertex must be same in both the private graphs. The generic Prim's algorithm is modified by taking two private graphs as input

1. for each u $\in$ V[G_1] , $\in$ V[G_2]
2. key[u] = ∞;
3. key[r]=0;
4. θ = V(G_1) AND V(G_2);
5. T = ϕ;
6. while ($\theta \neq \phi$)
7. {
8. u = Extract-Min(θ);
9. for each v $\in$ Adj[u] in G_1 and in G_2. Parties having graph G_1 and G_2 compute privately the length of the smallest edge among them.
10. $w(u,v)$ = min($w(u,v)_{G_1}$,$w(u,v)_{G_2}$)
11. if(u $\in \theta$ and w(u,v) $\leq$ key[v])
12. {
13. key[v] = w(u,v)
14. Add w(u,v) in T
15. T = T $\cup$ w(u,v)
16. }
17. }
18. return T

4 Complexity Analysis

For both the algorithms that are presented here, we can determine the total number of rounds, the total communication complexity and the computation complexity. By using the construction of Yao's protocol on circuits with m number of gates and n number of inputs it requires computational overhead as O(m+n), we need O(1) computational rounds and O(n) communications.

Complexity of privacy-preserving MSTK. For the analysis of our scheme we are assuming that the number of edges in E is n and l is the maximum edge length. Here we are considering input as length of edges in G_1 , G_2 and returns output as G = PPMSTK(gmin(G_1, G_2). For computing gmin sub protocol it holds O(n $log\ l$) number of gates. Therefore we can say that this approach needs rounds as $O(1)$ and $O(n\ log\ l)$ communication and computational.

Our scheme complexity is depending on the total number of minimum sub protocol iterations, in worst case it can be $n+n$. This is equal to the number of various edge lengths which are appearing in solution graph. At the stage of i^{th} iteration we are taking the minimum number of the two integers and constructing minimum spanning tree. This scheme requires $n-1$ iterations, at each iteration the minimum of two parties integer value is privately computed with $log\ l$ bits. As we are using Yao's method to find the minimum, it requires circuit representation with inputs as (n+n) $log\ l$ and gates as $O(log\ l)$. Both the parties are using oblivious transfers. Therefore 2 $log\ l$ oblivious transfers takes place in parallel.

We conclude the scheme which is used here will take number of rounds as constant that is $O(1)$ and total communication complexity is $O(n+n)(log\ l)$ simplified as $O(n\ log\ l)$ and total computational complexity is same that is $O(n\ log\ l)$.

Complexity of privacy-preserving MSTP. This is special case of minimum spanning tree comparing to PPMSTK it operates on adjacent minimum edges of vertex. Here the minimum spanning tree starts from a arbitrary starting root point identified by both private parties G_1 and G_2. At the time of terminating the algorithm θ must be empty. The performance of our approach is depends on min-heap i.e, θ.

For finding the shortest root paths the Prim's algorithm is absolutely similar to the Dijkstra's algorithm by considering the weights of edges as priorities rather than distances to source). It is every time connected because it the representation of the tree. We conclude that our scheme will take a constant number of rounds as $O(v)$ here v is number of vertices must be same in both the private parties, total number of oblivious transfers are $O(v\ (log\ v + log\ l))$ and total number of gates are $O(v\ (log\ v + log\ e))$ where l is length of the edge and e is set of edges.

5 Conclusions

We have presented privacy-preserving protocols in semi-honest model that describes two private parties to compute PPMSTK and PPMSTP on their private graphs. We proved that our protocols preserves privacy in semi-honest model. The future research is the same problem can be constructed using privacy-preserving protocols in malicious adversary model.

References

[1] A. Aly, E. Cuvelier, S. Mawet, O. Pereira, and M. Van Vyve. Securely solving simple combinatorial graph problems. In *Proc. of the 17th International Conference on Financial Cryptography and Data Security (FC'13), Okinawa, Japan*, volume 7859 of *Lecture Notes in Computer Science*, pages 239–257. Springer-Verlag, April 2013.

[2] D. Beaver. Foundations of secure interactive computing. In *Proc. of the 1991 International Cryptology Conference (CRYPTO'91), Berlin, Heidelberg*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. Springer-Verlag, August 1991.

[3] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Proc. of the 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'05), Chennai, India*, volume 3788 of *Lecture Notes in Computer Science*, pages 236–252. Springer-Verlag, December 2005.

[4] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, April 2000.

[5] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, July 1990.

[6] O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge University Press, May 2004.

[7] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM Conference on Theory of Computing (STOC'87), New York, USA*, pages 218–229. ACM, May 1987.

[8] Y. Lindell and B. Pinkas. A proof of yao's protocol for secure two-party computation. `https://eprint.iacr.org/2004/175`, [Online: accessed on August 20, 2018], July 2004.

[9] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of the 33rd ACM symposium on Theory of computing (STOC'01), Hersonissos, Greece*, pages 590–599. ACM, July 2001.

[10] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proc. of the 12th ACM-SIAM symposium on Discrete algorithms (SODA'01), Washington, D.C., USA*, pages 448–457. Society for Industrial and Applied Mathematics, January 2001.

[11] M. Naor and B. Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, October 2004.

[12] T. Toft. Secure data structures based on multi-party computation. In *Proc. of the 30th ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC'11), San Jose, California, USA*, pages 291–292. ACM, June 2011.

[13] A. C. Yao. Protocols for secure computations. In *Proc. of the 23rd IEEE Symposium on Foundations of Computer Science (SFCS'82), Chicago, Illinois, USA*, pages 160–164. IEEE, November 1982.

[14] A. C.-C. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Foundations of Computer Science (SFCS'86), Toronto, Ontario, Canada*, pages 162–167. IEEE, October 1986.

Author Biography

Koteswara Rao Ch received the B.Tech. degree from Acharya Nagarjuna University and M.Tech. degree in Computer Science and Engineering from Jawaharlal Nehru Technological University Kakinada and pursuing Ph.D. degree in National Institute of Technology, Tiruchirappalli, Tamil Nadu, India. Currently he is a Research Scholar in the department of Computer Science and Engineering in NIT Tiruchirappalli. His research interests include Cryptography, Multiparty Computation, Algorithms and Graph Theory. He is a member of CRSI.

Kunwar Singh received B.Tech. degree from IIT Delhi, M.Tech degree from Jawaharlal University, New Delhi and Ph.D degree from IIT Madras in 2015. Currently, he is Assistant Professor in Computer Science and Engineering Department at NIT Trichy, India since 2006. His area of research includes Public Key Cryptography, Identity-Based Encryption, Lattice Based Cryptography, Stream Ciphers, Multi-party Computation and Blockchain.
http://www.nitt.edu/home/academics/departments/cse/faculty/kunwar/