ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Maintaining Privacy and Secrecy in Untrusted Network with Stile

M. Anbu Chezhian[1], S.Ramalakhsmi[2]

ABSTRACT – Security and privacy issues have become critically important with the fast expansion of multi-agent systems. Most network applications such as pervasive computing, grid computing, and P2P networks can be viewed as multi-agent systems. sTile, a technique is proposed for distributing trust-needing computation onto insecure network systems which are open, anonymous, and dynamic in nature. sTile based implementation and empirically evaluate it on several physical networks of varying sizes, including the globally distributed system. Secure multiparty computational algorithm is used to ensure crystal neighbours do not learn each other's data. Multifactor authentication process is proposed to provide security in the untrusted network.

KEYWORDS – Privacy, Untrusted Network, sTile, Tile assembly model.

I. INTRODUCTION

The emergence of cloud computing is evolving the nature of computation. Instead of using private machines, users allow the cloud to maintain, manipulate, and safeguard their data. This evolution has allowed ubiquitous access to computation and data with higher availability and reliability than possible with personal machines and local servers. Simultaneously, this evolution has affected the meaning of the term privacy when referring to software systems. To ensure data remain private, not only must they be kept confidential from potential intruders, but also from the machines that execute computation on the data.

II. MOTIVATING EXAMPLE: ADDITION

To describe adding using sTile, we explain three separate elements of our solution: the addition tile assembly, the distribution process, and the source of privacy.

A. The Addition Tile Assembly

A tile assembly is a theoretical construct, similar to cellular automata. It consists of square tiles with static labels on their four sides. In Fig.1. Tiles can attach to one another or to a growing crystal of other tiles when sufficiently many of their sides match.

B. The Distribution Process

sTile uses the theoretical tile assembly to decompose a computation into small parts. Each small part represents a tile execution might be deployed on eight network nodes. Each node only deploys tiles of a single type, designated by the client machine. The client sets up a seed on the network by asking nodes that can deploy tiles of appropriate types to deploy instances of those types. Each node knows only the tile instances it is deploying and maintains references to the geometrically adjacent tile instances on other nodes. Next, tiles with an empty adjacent location coordinate with their crystal neighbour's to recruit matching tiles to attach this process uses a secure multiparty computation algorithm to ensure crystal neighbour's do not learn each other's data. Each of these steps replies on an algorithm that ensures the tiles are deployed uniformly randomly on the available nodes. Once the execution finishes, the tiles in the middle row report the solution to the client, indicating the node IDs of their crystal neighbour's, which the client uses to reconstruct the output.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

C. The Source of Privacy

Each tile instance is aware of only a single bit of the input, output, or intra computation data, and not of the bit's global location. An adversary may attempt to reconstruct the confidential data from the nodes it controls. For example, Fig. 1f shows an adversary that has compromised three nodes (2, 3, and 6), and now has access to the data in five tiles. However, this adversary can only tell that there are some 0 and 1 bits scattered throughout the input, the computation, and the output, but not how many and not their relative positions.

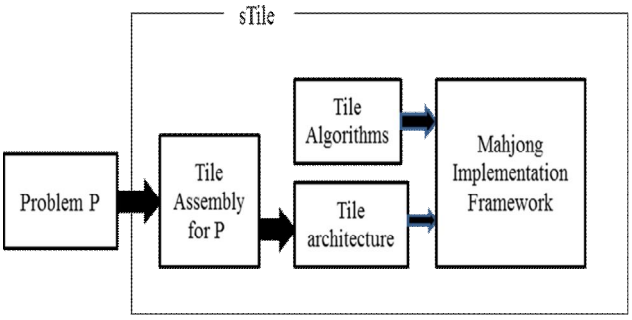

Fig.1. A high-level overview of sTile

III. STILE ARCHITECTURE AND ALGORITHMS

A sTile-based system is a software system that uses a network of computers to solve a computational problem. Intuitively, the network simulates a tile assembly: Each computer pretends to be a tile (or many tiles), and communicates with other computers to self-assemble a solution to a computational problem. Each computer deploys tile components, each representing a tile in a tile assembly, and facilitates the proper communication channels and algorithms to allow the tile component self-assembly. Thus, a tile architecture is based on a tile assembly; the software system employing that architecture solves the particular computational problem that the tile assembly solves.

A. Initializing Computation

The client computer initializes the computation by performing three actions: creating the tile type map, distributing the map and tile type descriptions, and setting up a seed crystal.

Creating the tile type map. A tile type map is a mapping from a large set of numbers (e.g., all 128-bit IP addresses) to tile types. It determines the type of tile components a computer with a given unique identifier (e.g., IP or MAC address) deploys. The tile type map breaks up the set of numbers into k roughly equal-sized regions, where k is the number of types of tiles in the tile assembly.

Distributing the map and tile descriptions. The client node distributes the tile type map and a short description of one tile type to a node that deploy that type, as determined by the tile type map. A tile type's description consists of the four tile component interfaces, which can be described using a few bits.

Creating a seed. The client is responsible for creating the first seed on the network through a fairly straightforward procedure. For each tile in the seed crystal described by the underlying tile assembly, the client selects a node that deploys that tile type and asks that node to deploy a tile.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

B. Discovery

The node discovery algorithm is central to sTile because initialization, replication, and recruitment all use it. The discovery operation, given a tile type, returns a uniformly random IP of some computer deploying tile components of that type. In Fig.2. Thus, every suitable computer has an equal chance of being returned, in the long run, which in turn guarantees that all nodes on the network perform a similar amount of computation. The algorithm uses a property of random walks to ensure uniform-randomness.

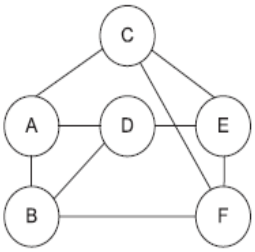

Fig.3. A network with six nodes. We assume that every node in our Underlying network has p network neighbors (here p ¼ 3).

C. Recruitment

The seed crystal grows into a full assembly by recruiting tile attachments. In a computational tile assembly (such as the assembly described in that solves 3-SAT), a tile component that has both an upper and a left crystal neighbor recruits a new tile to attach to its upper left. Fig. 3 indicates several places in a sample crystal where tile components are ready to recruit new tiles. A recruiting tile component X (highlighted in Fig. 3), for each tile type, picks a potential attachment node Y of that type from its node table, as described in Section III, and sends it an attachment request.

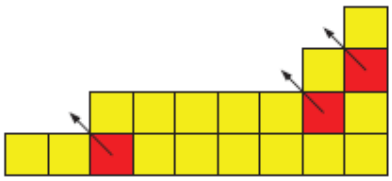

Fig.2. Tile components that have both an upper and a left crystal neighbor (highlighted in the diagram) can recruit new components to attach to their upper left.

D. Replication

Whenever network nodes have extra cycles they are not using for recruitment, they replicate the seed. Each node X uses its node table, as described in Section III, to find another node Y on the network that deploys the same type components as itself, and sends it a replication request. A replication request consists of up to two IP addresses (two 128-bit numbers) of X's crystal neighbors. X lets its crystal neighbors know that Y is X's replica (by sending Y 's IP to X's crystal neighbors). Those crystal neighbors, when they replicate using this exact mechanism, will send their replicas' IPs to Y . Thus, the entire seed replicates. Each component's replication is thus a three-step process: X sends a replication request to Y , Y replies to X, and X tells its crystal neighbors about Y . We analyze these steps in Section IV.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

E. Solution Reporting

One tile type, the black tile in Figs. 3 and 4e, includes in its encoding the identity (IP address) of the client. Recall that the black tile only attaches to a crystal when that crystal finds a solution. When that happens, the node deploying the black tile informs the client that the Boolean formula is satisfiable. While 3-SAT is a decision problem (i.e., the answer is either "yes" or "no"), the client may wish to also learn the Boolean assignment that satisfies the formula. To do so, the client may ask the node that notified it of the solution for its crystal neighbors' identities (IP addresses), and those for their crystal neighbors' identities, to reconstruct the entire crystal responsible for finding the solution. The client can then query the nodes that deploy the tiles encoding the assignment for their tile types. The cost of reconstructing the entire crystal is no more than contacting, and getting a response from each of the nodes deploying tiles in that crystal (310 tiles for the 3-SAT example from Fig. 4). However, since only part of the crystal is responsible for the assignment, it can be retrieved even more efficiently. To ensure privacy, all the relevant communication must be encrypted, and each involved node must verify the identity of the client. These requirements can be handled using standard public key encryption and authentication techniques, which we do not describe here.

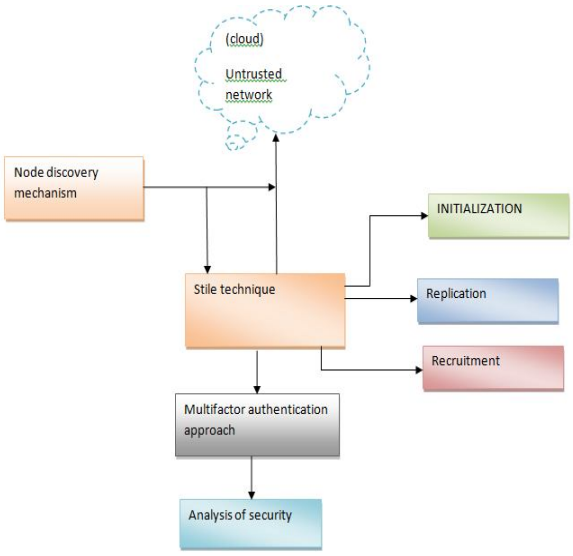

Fig.4. sTile Architecture design

IV. COMPUTATIONAL FEASIBILITY

To demonstrate that sTile is a feasible solution for building software systems that distribute computationally intensive problems on very large networks, we must show that 1) such systems' computational speed is proportional to the size of the underlying network, 2) such systems are robust to network delay, and 3) real-world-sized problems can be solved on real-world-sized networks in reasonable time.

A. sTile-Based Implementations

We have built and made public two Mahjong-based implementations, for 3-SAT and Subset Sum, as well as Simjong-based simulations of the same systems. Simjong is a Java-based discrete-event simulator with network-delay simulation capabilities. Simjong executes on a single machine and creates a user-specified number of virtual hardware Node components, each capable of deploying tiles. Simjong's network model allows for message delivery time to be

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

constant, chosen at random from some distribution, or proportional to the geographic distance between locations assigned to each virtual node. Simjong's network model is based on ns-2, simplified to abstract away the exact topology of the network.

***B.** Experimental Setup*

We use three distributed networks for our experimental evaluation: 1) a private heterogeneous cluster of 11 Pentium 4 1.5-GHz nodes with 512 MB of RAM, running Windows XP or 2000; 2) a 186-node subset of USC's Pentium 4 Xeon 3-GHz High Performance Computing and Communications cluster; and 3) a 100-node subset of PlanetLab, a globally distributed network of machines of varying speeds and resources that were often heavily loaded by several experiments at a time.

Our experimental goals were to verify sTile's scalability with respect to network size and robustness to network delay. Our experiments had three independent variables the number of nodes, the network communication speed between nodes, and the size of the NP-complete problem and one dependent variable the time the computation took to complete.

TABLE I

THE EFFECT OF DOUBLING THE NETWORK SIZE ON THE SYSTEM'S EXECUTION TIME. THE SPEEDUP RATIO IS THE FACTOR OF SPEED IMPROVEMENT OVER THE NETWORK OF HALF THE SIZE.

Network & Problem	Number of Nodes	Execution Time	Speed-up Ratio
Private Cluster	5	43.2 sec.	
𝔄	10	22.9 sec.	1.89
HPCC	93	220 min.	
𝔅	186	116 min.	1.90
PlanetLab	50	9.2 min.	
ℭ	100	4.8 min.	1.92
Simjong	125,000	8.7 hours	
	250,000	4.5 hours	1.93
𝔇	500,000	2.1 hours	2.14
	1,000,000	64 min.	1.97

C. Scalability

To verify that the speed of the computation is proportional to the number of nodes on the underlying network, for each of the three networks described above, we deployed Mahjong-based implementations on the entire network and on randomly selected halves of the network. We varied the size of the problem and measured the average time in which the implementations found the solution over 20 executions (except on PlanetLab, as explained above). We then also deployed Simjong on virtual networks of increasing size from 125,000 to 1,000,000 nodes (with a constant network delay of 100 ms for all packets).

D. Robustness to Network Latency

Intuitively, high-network latency should adversely affect the speed of sTile-based systems: If the tile attachments happen sequentially, the latency affects every attachment and greatly slows down the overall computation. This intuition holds for the addition example from Section 2. However, in the case of NP-complete computations, this intuition is false. In such computations, many of the sub computations (tile attachments) happen independently[7][9], in parallel. Each node in our experiments deployed millions of lightweight tiles, and whenever a sTile packet traveled

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

between nodes, those nodes handled other tiles rather than waiting idly for the network communication to arrive. As a result, the throughput of sTile-based systems is not affected by the network latency.

E. Efficiency

The final claim we address in demonstrating sTile's feasibility for industrial systems is that real-world-sized problems can be solved on real-world-sized networks in reasonable time. In particular, we posit that sTile-based systems can outperform existing privacy-preserving methods for solving NP-complete problems[12][14]. There are three ways to solve a highly parallelizable problem while preserving the data privacy: 1) on a large insecure network by using sTile, 2) on a single private computer, or 3) on a private network of trustworthy computers. We will first discuss the time needed to solve such a problem using the three methods in terms of the number of required operations, and then discuss the actual time necessary to solve problems.

$$\left(3i\frac{(3m+n)\lg n}{N}+5u\frac{3nm\lg^2 n}{N}\right)2^n \qquad (1)$$

$$2^n(n+3m)r. \qquad (2)$$

Now, suppose a user wishes to solve the 3-SAT instance on a single computer. That computer would need to examine 2n possible assignments, and check each n-variable assignment against the m clauses. Equation (2) describes the time this procedure would take using the most efficient available technique, assuming r is the amount of time each operation takes to execute: For each assignment, create a hash set containing the n literal-selection elements and check for each of the 3m literals whether the hash set contains that literal.

TABLE 2

COMPARISON OF EXECUTION TIME FOR SOLVING D AS MEASURED BY SIMJONG AND ESTIMATED BY (1)

Number of Nodes	Execution Time	
	Simjong	Estimate
125,000	8.7 hours	9.1 hours
250,000	4.5 hours	4.5 hours
500,000	2.1 hours	2.3 hours
1,000,000	64 min	68 min.

V. PRIVACY PRESERVATION

In this section, we formally argue that sTile systems preserve privacy. Specifically, we analytically argue that, as long as no adversary controls more than half the network, the probability of that adversary learning the input can be made arbitrarily low.

sTile's privacy preservation comes from each tile being exposed only to a few intermediate bits of the computation (see Fig. 4) and the tiles' lack of awareness of their global position. To learn meaningful portions of the data, an adversary needs to control multiple, adjacent tiles. We call a distributed software system privacy preserving if, with

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

high probability, a randomly chosen group of nodes smaller than half of the network cannot discover the entire input to the computational problem the system is solving[15][17]. (We will also discuss, at the end of this section, the probability of discovering parts of the input.) We argue that neither 1) a node deploying a single tile, nor 2) a node deploying multiple tiles can know virtually any information about the input; moreover, 3) controlling enough computers to learn the entire input is prohibitively hard on large networks.

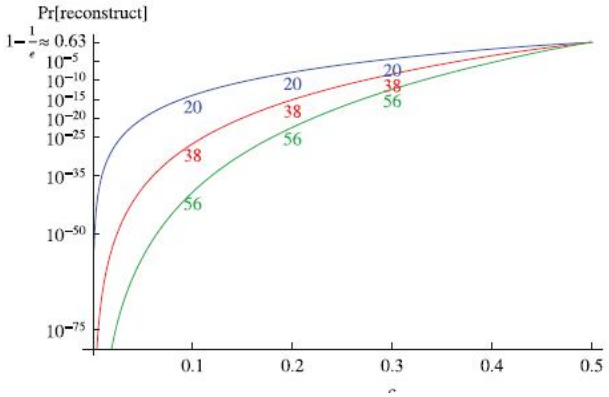

Fig. 5. The probability that an adversary controlling c fraction of the network can reconstruct an entire 20-, 38-, and 56-bit input.

VI. RELATED WORK

Distributing computation. The growth of the Internet has made it possible to use public computers to distribute computation to willing hosts. This notion focuses the underpinning of computational grids. Among systems that concentrate on distributed computation are BOINC systems (such as SETI@home and Folding@home), MapReduce , and the organic grid. A unique approach—FoldIt—uses the competitive human nature to solve the protein-folding problem[11]. These systems try to solve exactly the highly parallelizable problems toward which our work is geared, but unlike sTile, they do not preserve privacy.

Cloud privacy. Cloud computing has reemphasized the importance of data privacy[3][6], causing the emergence of numerous approaches for keeping data private on the cloud. Most such approaches concentrate on private data storage and user-authorized data retrieval and require some trusted agents whereas our work concentrates on preserving privacy during computation and requires no trusted agents.

Privacy-preserving computation. In classical (as opposed to quantum) computing, it is not possible to get help from a single entity in solving an NP-complete problem without disclosing most of the information about the input and the problem one is trying to solve[14]. Our approach avoids this shortcoming by distributing such a request over many machines without disclosing the entire problem to any small-enough subset of them.

VII. CONCLUSION

sTile distributes computation onto large, insecure, public networks in a manner that ensures privacy preservation, fault and adversary tolerance, and scalability. We presented a rigorous theoretical analysis of sTile and formally proved that the resulting systems are efficient and scalable, and that they preserve privacy as long as no adversary controls half of the public network.

We deployed two sTile implementations on several networks, including the globally distributed PlanetLab [36], to empirically verify

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

1. the correctness of sTile algorithms,
2. that the speed of sTile computation is proportional to the number of nodes,
3. that network delay has a negligible effect on the speed of the computation, and
4. that our mathematical analysis of the time needed to solve large problems on large networks is accurate. For networks larger than about 4,000 nodes, sTile outperforms optimized solutions that assume privately owned, secure hardware.

sTile explores the fundamental cost of achieving privacy through data distribution and bounds the extent to which a privacy-preserving system is less efficient than a nonprivate one. While that cost is not trivial, we have demonstrated that sTile-based systems execute orders of magnitude faster than homomorphic encryption systems, the alternative promising approach to preserving privacy.

VIII. ACKNOWLEDGEMENT

The authors thank Jae young Bang for his help deploying Mahjong-based implementations on PlanetLab. This material was based upon work supported by the US National Science Foundation (grant 0937060 to the Computing Research Association for the CIFellows Project and grants 0905665 and 1117593), and by Infosys.

REFERENCES

[1] L. Adleman, J. Kari, L. Kari, and D. Reishus, "On the Decidability of Self-Assembly of Infinite Ribbons," Proc. 43rd Ann. IEEE Symp. Foundations of Computer Science (FOCS '02), pp. 530-537, Nov. 2002.

[2] D.P. Anderson, "BOINC: A System for Public-Resource Comput-ing and Storage," Proc. Fifth IEEE/ACM Int'l Workshop Grid Computing (GRID '04), pp. 4-10, 2004.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Trans. Information and System Security, vol. 9, no. 1, pp. 1-30, Feb. 2006.

[4] D. Baker, "Foldit," http://fold.it, 2009.

[5] A. Balint, M. Henn, and O. Gableske, "A Novel Approach to Combine a SLS- and a DPLL-Solver for the Satisfiability Problem," Proc. 12th Int'l Conf. Theory and Applications of Satisfiability Testing (SAT '09), pp. 284-297, 2009.

[6] Y. Brun, "Solving NP-Complete Problems in the Tile Assembly Model," Theoretical Computer Science, vol. 395, no. 1, pp. 31- 46, Apr. 2008.

[7] Y. Brun, "Solving Satisfiability in the Tile Assembly Model with a Constant-Size Tileset," J. Algorithms, vol. 63, no. 4, pp. 151-166, 2008.

[8] Y. Brun, "Efficient 3-SAT Algorithms in the Tile Assembly Model," Natural Computing, vol. 11, no. 2, pp. 209-229, 2012.

[9] Y. Brun, G. Edwards, J. Young Bang, and N. Medvidovic, "Smart Redundancy for Distributed Computation," Proc. 31st Int'l Conf. Distributed Computing Systems (ICDCS '11), pp. 665-676, June 2011.

[10] Y. Brun and N. Medvidovic, "Mahjong: A sTile Framework for Distributing NP-Complete Computations Onto Untrusted Networks in a Trustworthy Manner," http://www.cs.umass.edu/brun/Mahjong, 2013.

[11] Y. Brun and N. Medvidovic, "Fault and Adversary Tolerance as an Emergent Property of Distributed Systems' Software Architectures," Proc. Second Int'l Workshop Eng. Fault Tolerant Systems (EFTS '07), pp. 38-43, Sept. 2007.

[12] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 5, pp. 847-859, May 2011.

[13] Wikipedia, "SETI@home," http://en.wikipedia.org/wiki/SETI@home, 2008.

[14] E. Winfree, "Simulations of Computing by Self-Assembly of DNA," Technical Report CS-TR:1998:22, California Inst. of Technology, Pasadena, CA, 1998.

[15] G.J. Woeginger, "Exact Algorithms for NP-Hard Problems: A Survey," Combinatorial Optimization - Eureka, You Shrink!, vol. 2570/2003, pp. 185-207, 2003.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On **Global Innovations In Computing Technology (ICGICT'14)**

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

[16] Z. Yang, S. Yu, W. Lou, and C. Liu, "P2: Privacy-Preserving Communication and Precise Reward Architecture for V2G Networks in Smart Grid," IEEE Trans. Smart Grid, vol. 2, no. 4, pp. 697-706, Dec. 2011.

[17] A.C.-C. Yao, "How to Generate and Exchange Secrets," Proc. 27th Ann. IEEE Symp. Foundations of Computer Science (FOCS'86),pp.162-167,Oct.1986.