Course work for PhD students aspiring to work in "Cryptology"

The following courses will be offered to the first semester Ph.D. students aspiring to work in the area of Cryptology and Security:

- AC: Advanced Cryptology
- STC: Selected Topics in Cryptology
- AT: Automata Theory
- QC: Quantum Computation
- DAA: Design and Analysis of Combinatorial Algorithms

The time table and details of each courses is given below.

Routine

	11:30 – 13:00	14:30 – 16:00
Monday	AC	DAA
Tuesday	STC	AT
Wednesday	QC	AC
Thursday	AT	DAA
Friday	QC	STC

5. Design and Analysis of Combinatorial Algorithms

A. **Description:** The course introduces the basics of computational complexity analysis and various algorithm design paradigms. The goal is to provide students with solid foundations to deal with a wide variety of computational problems, and to provide a thorough knowledge of the most common algorithms and data structures. After the course, a student should be able to analyze the asymptotic performance of algorithms, write rigorous correctness proofs for algorithms, and apply important algorithmic design paradigms and methods of analysis.

B. **Pre-requisites:** Basic Knowledge of Programming and Data Structures.

C. **Outline of the syllabus:**

(a) Foundations: Introduction, Motivation.

(b) Asymptotic complexity: informal concepts and formal notation, worst and average case analysis. Recurrence relations.

(c) Sorting: bubble sort, insertion sort, selection sort, merge sort, quick sort, stability and other issues with sorting.

(d) Data Structures: Hash Tables, Binary Search Trees, Red-Black Trees, B-Trees, Fibonacci Heaps.

(e) Elementary Graph Algorithms: BFS, DFS, Strongly Connected components, Topological sort.

(f) Shortest path Algorithms: unweighted and weighted; Single source shortest paths: Dijkstra; Minimum cost spanning trees: Prim's algorithm, Kruskal's Algorithm; Union-Find data structure.

(g) Divide and conquer: counting inversions, nearest pair of points; Priority queues, heaps, Priority queues, heaps, Dijkstra/Prim's revisited using heaps.

(h) Search Trees: Introduction, Traversals, insertions, deletions; Balancing.

(i) Greedy Algorithms: Interval scheduling, Proof strategies, Huffman coding.

(j) Dynamic Programming: weighted interval scheduling.

(k) String Matching: The Rabin-Karp algorithm, The Knuth-Morris-Pratt algorithm.

(l) Intractability: NP-Completeness, reductions, examples.

(m) Approximation Algorithms: The vertex-cover problem, The traveling-salesman problem, The set-covering problem, Randomization and linear programming, The subset-sum problem.

D. **Duration:** 45 hours (15 weeks, 3 hours per week).

E. **Learning outcome and the objective of the course:** Students who complete the course will have the ability to do the following: (i) apply knowledge of computing and mathematics to algorithm design, (ii) analyze a problem and identify the computing requirements appropriate for its solution, (iii) design, implement, and evaluate an algorithm to meet desired needs, (iv) apply mathematical foundations, algorithmic principles, and computer science theory to model and design computer-based systems in a way that demonstrates comprehension of the trade-offs involved in design choices, (v) apply design and development principles in the construction of software systems of varying complexity, and (vi) use current techniques, skills, and tools necessary for computing practice.

F. **References:**

(a) T. H. Cormen, C. E. Leiserson and R. L. Rivest: Introduction to Algorithms, Prentice Hall of India, New Delhi, 1998.

(b) A. Aho, J. Hopcroft and J. Ullman: The Design and Analysis of Computer Algorithms, A. W. L, International Student Edition, Singapore, 1998.

(c) J. Kleinberg, E. Tardos: Algorithm Design, Pearson Education, 2006.

G. **Assessment methodology:** Written and Programming assignments, Examinations.

H. **Pedagogic methodology:** Lectures, presentations, and Programming sessions.

The syllabus for this course is prepared by Laltu Sardar and Ritankar Mandal.