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

1. Advanced Cryptology

A. **Description:** Cryptology is concerned with the conceptualization, definition, and construction of computing systems that address security concerns. The design of cryptographic systems must be based on firm foundations. This course presents a rigorous and systematic treatment of the foundation issues: defining cryptographic tasks and solving new cryptographic problems using existing and new tools. The focus is given on the basic mathematical tools as well as some new advanced cryptographic tools and the advances of research using those tools.

B. **Pre-requisites:** Basic Cryptology, Number Theory.

C. **Outline of the syllabus**

1. **Foundation of Crypto:** Basic Complexity Results, BPP, Amplification Lemma of BPP, relation between NP and BPP, Notion of Negligible function and efficient Algorithms.

   ☐ One way Function: Definition of (strong and weak) One way function, construction of weak one way function, construction of strong one way function, Collections of one way functions, one way permutation, Collection of trapdoor one way permutations and examples, Some examples of not one way functions

   ☐ Hard Core Predicate: Definition of Hard core predicate, Examples, Construction of hard core predicates from one way function

   ☐ Pseudorandom Generator: Construction of PRG from Hard core predicate, Blum-blum Shub PRG, Some pathological examples of insecure PRG,

   ☐ Pseudorandom Function: GGM construction, Security proof of GGM PRF.

2. **Lattice Based Cryptography:** Basic definition of Lattice, Unimodular matrix, Fundamental region, Hermite Normal Form, GSO, Volume and Determinant of Lattice, Hadamard Inequality, Alternative definition of lattice as discrete subgroup, Minimum distance of lattice, Blitchfeld theorem, Convex body theorem, Minkowski's theorem, Minkowski's second theorem.

   ☐ Computational Problems on lattice: SVP, CVP, SIVP and their approximate version of the problem, LLL algorithm, Babai Nearest plane algorithm

   ☐ Application of LLL in Cryptography: Finding small root of univariate and bivariate polynomials.

3. **Multi Party Computation:**

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

E. **Learning outcome and the objective of the course**
FILL IT UP

F. **References**

(a) O. Goldreich: Foundations of Cryptography Vol 1.

(b) O. Goldreich: Foundations of Cryptography Vol 2.

(c) D. Boneh, V. Shoup: A Graduate Course in Applied Cryptography, online draft: `https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf`.

(d) J. Katz and V. Lyubashevsky: Lattice-Based Cryptography, CRC Press LLC, 2018.

(e) D. Micciancio, S. Goldwasser: Complexity of Lattice Problems: A Cryptographic Perspective, Kluwer, 2002.

(f) Recent Research Papers.

G. **Assessment methodology:** Mid-Semestral and End-Semestral Examination.

H. **Pedagogic methodology:** Lectures, presentations.

This course is proposed by Rana Barua and Avijit Dutta.

2. Selected Topics in Cryptology

A. **Description:** This course is aimed at introducing several advanced research topics in the field of Cryptology.

B. **Pre-requisites:** Basic Cryptology

C. **Outline of the syllabus:**

1. Provable Security: Revisiting PRF, PRP, SPRP, TBC, Lazy Sampling, Proof Models (Standard, Random Permutation, Ideal Cipher Model), Statistical Distance, Game-theoretic Technique, H-Coefficient Technique, Mirror Theory.
2. Design and Analysis of Authenticated Encryption: Motivation, Security Model, Generic Composition, Integrated AE, Application Specific desired properties of AE, Designing application specific AE.
3. Side-Channel Attacks: Concepts of leakage, Some basic attacks: timing and cache attacks, simple and differential power analyis, Fault attacks: concepts and motivation, Differential Fault attacks on AES, Fault based forgeries on CLOC, SILC.
4. White-box Cryptography: Motivation, Basic Concept, Construction of White Box AES.
5. Zero Knowledge Proof: The notion of Interactive Turing machines, Joint computations, Interactive Proof, IP Class, IP for Graph Non-Isomorphism, Definition of Zero knowledge proof, variants of ZKP, Auxiliary input ZKP, The notion of simulator, ZKP for graph isomorphism problem, Proof of any NP Language has a ZKP, Basic notion of bit commitment scheme, ZKP for Graph 3 coloring.

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

E. **Learning outcome and the objective of the course**
On completion of the course the student should have

F. **References**

(a) D. Boneh, V. Shoup: A Graduate Course in Applied Cryptography, online draft: `https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf`.

(b) Recent Research Papers.

G. **Assessment methodology:** Paper Presentation and Term Paper Examinations.

H. **Pedagogic methodology:** Lectures, Paper presentations, and Invited Talks.

This course is proposed by Nilanjan Datta, Avik Chakraborti and Avijit Dutta.

3. Automata Theory

A. **Description:** Automata theory deals with the logic of computation with respect to simple machines, referred to as automata. This is an abstract model of machines that perform computations on an input by moving through a series of states or configurations. At each state of the computation, a transition function determines the next configuration on the basis of a finite portion of the present configuration. As a result, once the computation reaches an accepting configuration, it accepts that input. Through this course, the students should be able to understand how machines compute functions and solve problems and more importantly, what it means for a function to be defined as computable or for a question to be described as decidable.

B. **Pre-requisites:** High-school Mathematics.

C. **Outline of the syllabus:**

1. **Automata and Languages:** Finite Automata, Regular Languages, Reg- ular Expressions, Deterministic and Non-deterministic Finite Automata, Minimization of Finite Automata, Closure Properties, Kleene's Theorem, Pumping Lemma and Its Applications, Myhill-Nerode Theorem and Its uses, Decision Algorithms for regular languages; Context-free Grammers, Context-free Languages (CFL), Chomsky Normal Form (CNF), Closure Properties, Pumping Lemma for CFL, Push Down Automata, Acceptance by empty stack and final state, Decision Algorithms for CFL.
2. **Computability:** Turing Machine and variants, Computable Functions, Primitive and Recursive Functions, Universality, Halting Problem, Recursive and Recursively Enumerable Sets, Diagonalisation, Reducibility.
3. **Introduction to Complexity:** Discussions on Time and Space Complexities, P and NP, NP-completeness, Cook's Theorem, other NP-Complete Problems, PSPACE.

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

E. **Learning outcome and the objective of the course:** The course focuses on three traditionally central areas of the theory of computation: automata, computability and complexity. A student learning this course, should be able to do the following:

(a) Mathematically model machines that perform some dedicated computation (automata theory). These computations may involve some simple operations such as controller of an elevator as well as involved operations like a compiler or hardware design.

(b) Classify problems as easy ones and hard ones (from the complexity theory),

(c) Classify problems as solvable or not (in the direction of theories of computability)

F. **References:**

1 M. Sipser: Introduction to The Theory of Computation, PWS Pub. Co., New York, 1999.

2 J. E. Hopcroft, J. D. Ullman and R. Motwani: Introduction to Automata Theory, Languages and Computation, Addison- Wesley, California, 2001.

3 M. D. Davis, R. Sigal and E. J. Weyuker: Complexity, Computability and Languages,Academic Press, New York, 1994.

4 H. R. Lewis and C. H. Papadimitriou: Elements of The Theory of Computation, Prentice Hall, Englewood Cliffs, 1981.

5 M. R. Garey and D. S. Johnson: Computers and Intractability: A Guide to The Theory of NP Completeness, Freeman, New York, 1979.

G. **Assessment methodology:** Examinations: Midterm (40%), Endterm (60%).

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

This course is proposed by Rana Barua and Nilanjan Datta.

4. Quantum Information and Cryptography

A. **Description:** Objective of the course is to build quantum-preparedness for post quantum era. Due to Grover, Shor and Simon algorithms classical private and public key cryp- tographic security might be compromized totally or partially. By Post Quantum Cryptography, we generally mean the Public Key Cryptography which are resistant against quantum adversary, an adversary having unbounded power of computation. NIST has already started this effort. Another avenue is Quantum Cryptography. In this direction, to provide the students an overview about this paradigm, we organize the syllabus as follows.

B. **Pre-requisites:** Mathematics at undergraduate level, High-school level programming, Basic knowledge on Cryptology

C. **Outline of the syllabus:**

1. Introduction to Quantum Information
   (a) States, Operators, Measurements
   (b) Quantum Entanglement: Quantum Teleportation, Super-dense coding
   (c) Quantum gates and circuits
2. Quantum Algorithms: Deutsch-Jozsa, Simon, Grover, Shor, and their cryptanalytic implications
   (a) Implication of Grover's and Simon's algorithms towards classical symmetric key cryptosystems
   (b) Implication of Shor's algorithm towards factorization and Discrete Logarithm based classical public key cryptosystems
3. Quantum True Random Number Generators (QTRNG)
   (a) Detailed design and issues of quantumness
   (b) Commercial products and applications
4. Quantum key distribution (QKD)
   (a) BB84, Ekert, Semi-Quantum QKD protocols and their variations
   (b) Issues of Device Independence
   (c) Commercial products
5. Other cryptologic issues, such as Quantum secret sharing and multiparty computation
6. Introductory topics in Post-Quantum Cryptography

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

E. **Learning outcome of the course:** The following are expected from the students after completion of the course.

- Basic understanding about Quantum Information and Computation.
- Students should write and run program(s) in IBM quantum computers.
- Students should understand how to implement Grover key recovery attack on classical symmetric ciphers.
- Getting an overall understanding about Quantum Key Distribution and Secret Sharing protocols.
- To obtain the flavour of quantum supremacy over classical computation.

F. **References**

1 M. A. Nielsen and I. L. Chuang: Quantum Computation and Quantum Information, Cambridge University Press

2 P. Kaye, R. Laflamme, and M. Mosca: An Introduction to Quantum Computing, Oxford University Press, New York

3 Presskil Lecture notes: Available online: http://www.theory.caltech.edu/ preskill/ph229/

4 N. David Mermin: Quantum Computer Science, Cambridge University Press

5 D. Unruh: Quantum Cryptography, Available online: https://courses.cs.ut.ee/all/ MTAT.07.024/2017_fall/uploads/

6 NIST Post Quantum Cryptography, Available online: https://csrc.nist.gov/projects/ post-quantum-cryptography/round-2-submissions

G. **Assessment methodology:** There will be a Midterm (40%) and a Final (60%).

H. **Pedagogic methodology:** The course will be taught through lectures, assigned readings and class discussion.

The syllabus for this course is proposed by Dr. Arpita Maitra.

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.