Mukesh Tiwari

Education

2016–2020 **PhD, Computer Science**, *Australian National University*, Canberra, Australia

2004–2009 **Integrated Postgraduate**, *Indian Institute of Information Technology & Management*, Gwalior, India

PhD thesis

Title *Formally Verified Verifiable Electronic Voting Scheme*

Supervisor Dirk Pattinson

Description We focussed on three main challenges posed by electronic voting: correctness, privacy, and verifiability. We addressed correctness by using a theorem prover to implement a vote-counting algorithm, privacy by using homomorphic encryption, and verifiability by generating a independently checkable scrutiny sheet. Our work had been formalised in the Coq theorem prover.

Employment

2021-2023 **Senior Research Associate**, *University of Oxford*, Oxford, United Kingdom

I am working on connecting symbolic models of cryptographic protocols and their verified implementations using session types. The goal is to obtain a mathematically proven correct distributed implementation that mimics the real-world situation.

2021-2023 **Senior Research Associate**, *University of Cambridge*, Cambridge, United Kingdom

I worked on formalising a networking-protocol framework based on semiring algebraic structure in the Coq theorem prover. The goal was to develop a mathematical proven correct framework so that a protocol designer could assess the properties of their protocols using my framework.

2020-21 **Research Associate**, *University of Melbourne*, Melbourne, Australia

I worked with Toby Murray on *Security Concurrent Separation Logic*. The aim was to mathematically reason about memory safety and information flow property of concurrent programs written in C.

2018-20 **Tutor**, *Australian National Univeristy*, Canberra, Australia

I was a tutor for a first year logic course and Haskell programming course. My role was to help students understand the concepts, clearing their doubts, and assisting them in homework.

2013–2015 **Lecturer**, *International Institute of Information Technology*, Bhubaneswar, India

This role was primarily teaching focussed, and the courses I taught were *C programming, Java Programming, Web Technologies, Compiler Design,* and *Cryptography*. In addition, every year I supervised two master's students in their final year project.

2012–2013 **Haskell Developer**, *Parallel Scientific*, Colorado, USA

In this role, my primary job was research and prototype high performance software programs, mainly linear algebra algorithms written in Haskell.

☎ *+44-7824648138*
✉ *mt883@cam.ac.uk, mukesh.tiwari@anu.edu.au, mukeshtiwari.iiitm@gmail.com*
🌐 *mukeshtiwari.github.io/* • in *mukesh-tiwari-3609486/*
mukeshtiwari

2009–2012 **Technical Assistant**, *Government of India*, Kolkata, India

I worked as a developer for automating the day-to-day job, including enforcing the security policies of the organisation.

2008–2008 **Summer Intern**, *Arcelor-Mittal, Research & Development Technological Centre*, Avilés, Spain

I worked on formalising many business requirements into a linear programming problem and wrote a custom interface that interacted with Arcelor-Mittal's in-house linear programming solver.

Conference Publication

[1] Toby Murray, Mukesh Tiwari, Gidon Ernst, and David A. Naumann. Assume but Verify: Deductive Verification of Leaked Information in Concurrent Applications. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS '23, Copenhagen, Denmark, 26-30 Nov. `https://github.com/mukeshtiwari/IFMachine/`. (co-developer with Toby Murray and Gidon Ernst. In this work, I formally verified the case studies: differentially private location-server, federated machine learning server, aution-server, and email-server in SecureC; project duration: 1.6 years).

[2] Thomas Haines, Rajeev Goré, and Mukesh Tiwari. Machine-checking Multi-Round Proofs of Shuffle:Terelius-Wikstrom and Bayer-Groth. 32nd USENIX Security Symposium (USENIX 2023), Anaheim, California, USA, August 9-11, 2023. `https://github.com/mukeshtiwari/secure-e-voting-with-coq`. (co-developer with Thomas Haines. I proved facts related to zero-knowledge-proof in the Coq theorem prover; project duration: 2 years).

[3] Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. Verifpal: Cryptographic Protocol Analysis for the Real World. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, Progress in Cryptology - INDOCRYPT 2020, pages 151–202, Cham, 2020. Springer International Publishing. (co-developer with Georgio Nicolas. I worked on proofs related to Verifpal model in Coq; project duration: 8 months)

[4] Thomas Haines, Rajeev Goré, and Mukesh Tiwari. Verified Verifiers for Verifying Elections. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, page 685–702, New York, NY, USA, 2019. Association for Computing Machinery. `https://github.com/mukeshtiwari/secure-e-voting-with-coq`. (co-developer with Thomas Haines. I worked on efficient finite field arithmetic, required for efficient zero-knowledge-proof validation of well-formedness of a ballot; project duration: 1 year)

[5] Thomas Haines, Dirk Pattinson, and Mukesh Tiwari. Verifiable Homomorphic Tallying for the Schulze Vote Counting Scheme, In Verified Software: Theories, Tools, and Experiments. Springer, 2019. `https://github.com/mukeshtiwari/EncryptionSchulze/tree/master/code/Workingcode` (lead developer, project duration: 2 years)

[6] Milad K. Ghale, Rajeev Goré, Dirk Pattinson, and Mukesh Tiwari. Modular Formalisation and Verification of STV Algorithms. In Robert Krimmer, Melanie Volkamer, Véronique Cortier, Rajeev Goré, Manik Hapsara, Uwe Serdültt, and David Duenas-Cid, editors, Electronic Voting, pages 51–66, Cham, 2018. Springer International Publishing. `https://github.com/mukeshtiwari/Modular-STVCalculi`. (co-developer with Milad K. Ghale. I proved some of the critical theorems, required for code extraction; project duration: 8 months)

[7] Lyria Bennett Moses, Rajeev Goré, Ron Levy, Dirk Pattinson, and Mukesh Tiwari. No More Excuses: Automated Synthesis of Practical and Verifiable Vote-Counting Programs for Complex Voting Schemes. In Robert Krimmer, Melanie Volkamer, Nadja Braun Binder, Norbert Kersting, Olivier Pereira, and Carsten Schürmann, editors, Electronic Voting, pages 66–83, Cham, 2017. Springer International Publishing. `https://github.com/mukeshtiwari/formalized-voting/tree/master/SchulzeOCaml` (lead developer, project duration: 8 months)

[8] Dirk Pattinson and Mukesh Tiwari. Schulze Voting as Evidence Carrying Computation. In Mauricio Ayala-Rincón and César A, Muñoz, editors, *Interactive Theorem Proving*, pages 410–426. Cham, 2017. Springer International Publishing. `https://github.com/mukeshtiwari/formalized-voting/blob/master/paper-code` (lead developer, project duration: 1 year)

[9] Mukesh Tiwari, Karm V. Arya, Rahul Choudhari, and Kumar S. Choudhary. Designing Intrusion Detection to Detect Black Hole and Selective Forwarding Attack in WSN Based on Local Information. In 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology, pages 824–828, Nov 2009. (lead developer; project duration: 1 year)

[10] Rahul Choudhari, Karm V. Arya, Mukesh Tiwari, and Kumar S. Choudhary. Performance Evaluation of SCTP-Sec: A Secure SCTP Mechanism. In 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology, pages 1111–1116, Nov 2009. (co-developer with Rahul Choudhari; project duration: 1 year)

Workshop Publications

[1] Mukesh Tiwari and Dirk Pattinson. Machine Checked Properties of the Schulze Method. 7th Workshop on Hot Issues in Security Principles and Trust 2021.

[2] Mukesh Tiwari. Towards Leakage-Resistant Machine Learning in Trusted Execution Environments. Program Analysis and Verification on Trusted Platforms (PAVeTrust) Workshop 2021.

[3] Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. Verifpal: Cryptographic Protocol Analysis for the Real World. Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop 2020.

Work in Progress

[1] Formally Verified Verifiable Group Generators. In this work, we develop a formally verified algorithm that can be used to bootstrap a democratic election (sole author) (work in progress). https://github.com/mukeshtiwari/Formally_Verified_Verifiable_Group_Generator.

[2] Modelling Networking Protocols Mathematically. In this work, we develop a formally verified framework that a network protocol designers can use to verify the properties of their protocols (joint work Timothy Griffin. In this work, I formally verified generalised graph algorithm on semiring algebra in the Coq theorem prover). (ongoing and planning to submit to CAV 2024). https://github.com/mukeshtiwari/Semiring_graph_algorithm

[3] An Algebraic Framework for Multi-Objective Optimisation. In this work, we develop a formally verified framework in the Coq theorem prover that can be used to model various multi-objective optimisation problem as a graph algorithm in the Semiring framwork. (joint work Timothy Griffin but I am leading the project). (ongoing). https://github.com/mukeshtiwari/Formally-Verified-MultiObjective-Optimisation

[4] Theorem Provers to Protect Democracies. In this work, we are formalising all the cryptographic components written in Java of SwissPost in the Coq theorem prover. Our goal is to replace the SwissPost Java implementations (https://bit.ly/3EODmnF) with mathematically proven correct Coq implementations to write an independent verifier for the scrutiny sheet of elections conducted by Swiss Post software programs (sole author) (work in progress and planning to submit to IEEE S&P 2024). https://github.com/mukeshtiwari/Dlog-zkp.

[5] Machine Checked Properties of the Schulze Method. In this work, we are formally verifying all the (social choice) properties of the Schulze method. (lead developer, joint work with Dirk Pattionson) (work in progress). https://github.com/mukeshtiwari/Schulzeproperties.

Community Service (Reviewer)

[1] ACM Transactions on Privacy and Security

[2] Sadhana - Academy Proceedings in Engineering Sciences

[3] IEEE Security & Privacy

Skills

Coding	Coq, Lean, Haskell, OCaml, Python, C, Racket
Language	Hindi, English

Awards

HDR Fee Remission Merit Scholarship

ANU PhD Scholarship (International)

Full Scholarship to attend DeepSpec Summer School 2018, Princeton University

Travel Scholarship to attend Marktoberdorf Summer School 2019

References

○ Dirk Pattinson, Research School of Computer Science, Australian National University, Canberra, dirk.pattinson@anu.edu.au

○ Toby Murray, School of Computing and Information Systems, University of Melbourne, Melbourne, toby.murray@unimelb.edu.au

○ Timothy Griffin, Computer Laboratory, University of Cambridge, Cambridge, tgg22@cam.ac.uk

○ Thomas Haines, Research School of Computer Science, Australian National University, Canberra, thomas.haines@anu.edu.au

Mukesh Tiwari
Oxford, United Kingdom
📱 *+447824648138*
✉ *put here oxford email*

To
The Hiring Committee
University of Illinois, Urbana-Champaign, USA

Application for the Faculty (Open Rank)-Electrical and Computer Engineering

Dear Hiring Committee,

My name is Mukesh Tiwari and I am a senior research associate at the University of Oxford, UK with expertise in formal methods, cybersecurity and privacy, and social choice theory. I am writing to apply for the job **Assistant Professory (Faculty (Open Rank)-Electrical and Computer Engineering)**. I have extensive research experience in formal verification (Coq theorem prover), electronic voting, and cryptography, and my research experience makes me a valuable candidate for this post.

My research touches the lives of common people and solves real-world problems that matter to democracies and common people. For example, my paper (i) **Assume but Verify: Deductive Verification of Leaked Information in Concurrent Applications**, accepted in ACM CCS, develops a theory using the information-flow security principals for processing sensitive data –ethnic origin, political opinions, health-related data, and biometric data– of common people in secure enclave, e.g., Intel SGX, Arm TrustZone, etc. Moreover, we demonstrate the usability of our method by developing non-trivial case studies that handles sensitive data accompanied by the machine-checked mathematical proofs that none of them have unintended side-channel data leakage; (ii) **Machine-checking Multi-Round Proofs of Shuffle: Terelius-Wikstrom and Bayer-Groth**, published in USENIX Security, mathematically establishes a critical piece of code in the SwissPost voting software –used in legally binding elections in Switzerland– is correct (and debunks a decade old myth of the cryptographic community that Terelius-Wikstrom method is zero-knowledge proof. We have formally proved in the Coq theorem prover that it is a zero-knowledge argument and not a zero-knowledge proof); (iii) **Verifiable Homomorphic Tallying for the Schulze Vote Counting Scheme**, published in VSTTE, not only developes a publicly verifiable method to count encrypted ballots for a complex voting method but it is also proven correct in the Coq theorem prover to ensure that there is no gap between the pen-and-paper proof and the actual implementation; (iv) **Verified Verifiers for Verifying Elections**, published in ACM CCS, develops a mathematically proven correct tool in the Coq theorem prover to verify the elections conducted by the International Association for Cryptologic Research. We have used our tools to verify the integrity of IACR elections; (v) **Modular Formalisation and Verification of STV Algorithms**, published in E-Vote, develops a mathematically proven correct tool in the Coq theorem prover. We have used this tool to verify the results of Australian Senate election; (vi) **Verifpal: Cryptographic Protocol Analysis for the Real World**, published in INDOCRYPT, develops a tool that can used to model real work cryptographic protocol, and Verifpal has been used by many researchers to model security and privacy aspect of digital contact tracing during COVID, etc. At Cambridge, I developed a mathematically proven correct tool in the Coq theorem prover that can be used by networking researchers to model networking-protocols in the abstract setting of semirings (and we are in the process of submitting our paper in CAV 2024). Currently at Oxford, I am exploring the avenues to combining security protocol with their implementations using session types. Our goal is to produce a formal executable model of real-world (distributed) applications, e.g., **Signal** chat messenger, etc. In a nutshell, all my research so far has an impact on the lives on common people and researchers.

Although, as a person, I am slightly introvert, but I firmly believe interdisciplinary research is the key to solve challenging problem pertaining to society. Therefore, I like to chat and work with diverse set of researchers, which is evident from my projects involving myriad of concepts, e.g., formal method, cryptography, voting, social choice, separation logic, information-flow security, graph theory, session types, etc.

Your Sincerely,

Mukesh Tiwari

Research Statement

Mukesh Tiwari

My work aims to build **correct software** using the Coq theorem prover. I focus on formal verification of computer programs used in elections, cryptography, networking, and computational social choice theory. Below are some potential projects that I would like to focus, if given the position.

1 Future Work

- I would like to focus on formally verified cryptographic primitives used in verifiable computing, multi-party computation (MPC), and blockchain. The recent development in the area of zero-knowledge-proofs, in particular zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK), has led to a exponential growth in many useful privacy preserving applications in verifiable computing, MPC, etc. Most of these applications, however, are written in unsound language (Rust/Python/C/Java) and contain bugs, lack security proof, and therefore using formal verification to improve their correctness and security would bolster the confidence of a user of these applications. In addition, zk-SNARK has massive potential in electronic-voting and machine learning.
  - The current practice to verify the integrity of an election conducted electronically is to recompute the whole count on the publicly available (electronic) ballots, but this excludes many voters from verifying the integrity of the election because they do not posses a powerful enough computing device. Many of zk-SNARKs produce a very small size proof for the integrity of a computation regardless of the computation data. Therefore, if we deploy zk-SNARK in electronic-voting, it would increase the number of scrutineers because anyone, including the mobile devices holders, can verify the integrity of an election by checking these small size proofs.
  - In recent years, machine learning models are getting bigger and bigger and cannot be trained using a normal computer. Therefore, most of them are trained in cloud, but this raises the question if the cloud has used the right set of data to train them model or not. One way to solve this problem is to use verifiable computing and force the cloud to generate a short proof (zk-SNARK) with the trained machine learning model. This short proof can be then checked by any independent third party to attest the integrity of training.
- At the University of Cambridge, I worked on formalisation of various graph algorithms using the Coq theorem prover in *semiring* (algebraic structure). In our framework, depending on concrete instantiation of semiring operators, the same algorithm can compute shortest path, longest paths, widest paths, multi-objective optimisation, and many more. In addition, I also worked on formalisation of theory of algebraic reductions. However, the focus of this project was path problems but in future I would like to formally verify generalised (semiring) algorithms related to data flow analysis of imperative programs (Newtonian program analysis), Petri nets, neurosymbolic programming, etc., in the Coq theorem prover.
- At the University of Melbourne, I worked on security concurrent separation logic for formally reasoning about the information flow security in concurrent programs. I used *SecureC*, a tool developed at the university of Melbourne, to formalise an email server, an auction server, a location server,

and a differentially private gradient descent algorithm. All these works were proven to leak no sensitive information to attackers, assuming that the compiler respects all assumption and soundness of implementation of *SecureC*. However, *SecureC* is very limited in terms of expressibility and tooling and therefore it cannot be used to model a real world case study. I would like to work on adding information flow support in Iris –a Coq library– with my co-author Toby Murray.

- In one of my recent project, me with my co-authors –after 1.5 years of painstaking effort– formalised a critical piece of code (mix-network) in the SwissPost –used in legally binding elections in Switzerland– in the Coq theorem prover and extracted an OCaml code from our formalisation against which the Java code of SwissPost can be tested. In the process, we found a flaw in a decade old cryptographic proof. Given my expertise in formal verification, I would like with my PhD students and postdocs researchers to explore and formalise other piece of code used in public domain, e.g., machine learning models and code deployed in self-driving cars, decision-making algorithms, etc.

2 Past Work

My PhD research was focused on verifying electronic voting, specifically vote-counting schemes, in the Coq theorem prover. The goal was to bring three important ingredients (i) correctness, (ii) privacy, and (iii) verifiability of a paper-ballot election to an electronic setting (electronic voting). In my thesis, I demonstrated the correctness of the Schulze method by implementing it and proving its correctness in the Coq theorem prover. The Schulze method is a preferential (ranking) voting method where voters rank the participating candidates according to their preferences. It is one of the most popular voting method amongst the open-source projects and political groups[1]. In addition, my implementation ensured (universal) verifiability by producing a scrutiny sheet with the winner of an election. The scrutiny sheet contained all the data related to the election that could be used to audit the election independently. The extracted OCaml code from this formalisation, however, was very slow, so I wrote another fast implementation, proven equivalent to the slow one, capable of counting millions of ballots.

In both formalisation, I assumed that (preferential) ballots were in plaintext, i.e., ranking on every ballot was in a (plaintext) number. Preferential ballots, however, admit "Italian" attack. If the number of participating candidates are significantly high in a preferential ballot election, then a ballot can be linked to a particular voter if published on a bulletin board. A coercer demands the voter to mark them as first and for the rest of candidates in a certain permutation. Later, the coercer checks if that permutation appears on the bulletin board or not. In order to avoid this attack on the Schulze method, I used homomorphic encryption to count the (encrypted) ballots, without decrypting any individual ballot. Moreover, I addressed verifiability by generating a scrutiny sheet (certificate) augmented with zero-knowledge-proofs for various claims, e.g., honest decryption, honest shuffle, during the counting. Finally, I wanted to develop to formally verified scrutiny sheet checker for encrypted ballots Schulze election, but due to the lack of time[2] I worked on a scrutiny sheet checker for a simple approval voting election, International Association of Cryptologic Research (IACR) election. In addition, I was involved in formalisation of single transferable vote, used in the Australian Senate election.

My lab will be a diverse place where students and researchers from formal verification, cryptography, political science, social science, social choice theory will interact, discuss, collaborate on the ideas that matters to democracies and common people.

[1] `https://en.wikipedia.org/wiki/Schulze_method#Users`

[2] PhD duration is 3.5 years in Australia

Teaching Statement

Mukesh Tiwari

My teaching philosophy is not to immediately reach a solution but to develop a thinking process (problem solving mindset) that leads to the solution. I believe every student is different and has a unique style of learning, and my role is to help them find and hone their style. When I started teaching as an assistant professor at the International Institute of Information Technology (IIIT), Bhubanesware, India[1], my single biggest challenge was keeping the students engaged in my class, especially the first year students in C programming course. At the IIIT, I taught C programming to first year students, Compiler Design and Java programming to third year students, and Cryptography to final year (4th year) students. In each course, every single problem, more or less, boiled down to keeping the students engaged in a topic. In order to keep them engaged in a class, I took a Coursera course on learning[2] and read many academic articles and non-academic articles about effective learning. I tried some of the techniques suggested in the Coursera course and academic and non-academic articles in my classes. Below I describe my experiences with teaching and efforts to engage my students.

1 Setting Clear Goals

In every course, I started with the end goal of the course. For example, in C programming course, I told my students that by the end of this course they would be able to write simple C programs, e.g., calculator, validating a debit card, and other simple real-world problems. The rationale was to show a vision to excite them for learning and instill the feeling of empowerment, from being a consumer to being a developer of software programs.

2 Start with Why

One thing that I learnt by teaching for 3 years is that if you want a concept to stick in someone's mind, then start with a *why*[3]. For example, when I introduced functions in C programming course, I wanted to convey the need of functions. Therefore, I started the class by asking them to write a program, using pen and paper, of **factorial of a number** n ($n!$). Every one was comfortable with loops, so it was quick. I then asked them to write another program: k^{th} **power of the factorial of** n ($(n!)^k$). In this assignment, some added an outer loop to cover the factorial computation, and some added another loop after the factorial computation (stored the factorial computation result and used it in later computation). At this point, I asked the class how one could write a program

[1] It is a small technical school

[2] `https://www.coursera.org/learn/learning-how-to-learn`

[3] I learnt this idea by reading the book *Start With Why* by Simon Sinek.

of **factorial of the k^{th} power of the factorial of** n $(((n!)^k)!)$. The class slowly realised that they needed to duplicate a lot of code. I then introduced functions and showed them the solution of the previous problem by function composition.

3 Catching my Mistakes

To promote active participation, at the beginning of every class I would announce that on a few occasions I would make deliberate mistakes in solving a problem, and the goal of the class was to catch me on the spot. If the class succeeded, they received class participation marks, otherwise I told them my mistake and no one received any marks. In every class, especially in programming, first I would teach a topic, and later I would solve some problems on a blackboard, related to the topic. It was during the problem solving where I committed deliberate mistakes. It increased the class participation by an order of magnitude[4]. In every single feedback that I got during my 3 years of teaching, almost everyone appreciated this idea of making deliberate mistakes.

4 Potential Courses

I feel qualified to teach most of the computer science courses because of my background in computer science, but my natural preference is the courses close to my research area, e.g., theorem proving, cryptography, logic and its applications in computer science, discrete mathematics, algorithms and data structures, introductory programming course (C/C++/Java/Python/Haskell/OCaml), foundation of computing, etc. In addition, I would also like to design a course to teach various voting methods used around the world and their advantages and disadvantages from a social choice theory perspective. Apart from these topics, I will be more than happy to teach other courses, given enough preparation time, at introductory and intermediate levels, e.g., type theory, theory of programming languages, networking, operating systems, databases, etc.

Finally, I would like to emphasize the my teaching philosophy is not to immediately reach a solution but to develop a thinking process (problem solving mindset) that leads to the solution.

[4] I learnt this from advertising agencies where they write some ads wrong intentionally to grab the attention.

Diversity, Equity and Inclusion

Mukesh Tiwari

Diversity and inclusion matters a lot to me because of my own background (born in a farming community in a remote Indian village with no school). I believe everyone is capable of achieving everything, all they need is resource and support. Therefore, I firmly believe that all institutions in a democracy, including universities should strive to build community of individuals with diverse backgrounds and life experiences, free of discrimination based on racial and ethnic origin, gender identity, sexual orientation, social economic status or religious belief.

In the past, when I was working as an assistant professor in International Institute of Information Technology (IIIT) Bhubanesware, I participated in teaching workshops, organised by the student body of IIIT, to teach the children of migrant workers, working as a construction worker in IIIT. In addition, whenever I visit my village, I teach physics, mathematics, and basic computer literacy course to all the children, coming from economically underdeveloped community, in evenings. However, I lack funds to buy computers for everyone and therefore, I have been recently trying to encouraging people to donate their computers. My long term goal is to lower the gap of digital divide and make the children from economically underdeveloped community literate in computer education and employable in computer industry to break the cycle of poverty. I am committed to promoting diversity, equity, and inclusion (DEI) in my lab by hiring students from diverse backgrounds.

I am also a staunch supporter of a robust and transparent democracy, and one of my future goal is to end the first-past-the-post (FPTP) voting method in India by challenging the Election Commission of India in the Indian Supreme Court. FPTP is an unsatisfactory method to elect the winner of an election and it is actively exploited by the political parties in India (sometimes, a party with even 33% votes wins the state election). Therefore, elections in India is no longer about the voters but about strategic manipulation. Consequently, in recent elections, many voters are boycotting the elections by spoiling their ballot using NOTA (none of the above).

Leadership Philosophy

Mukesh Tiwari

I have one core value as a leader: help and support people to achieve their best. Everything else hinges around it. For example, when I was a teacher in a small technical university in India, my goal was not to immediately reach a solution but to develop a thinking process (problem solving mindset) that leads to the solution. I believe every student is different and has a unique style of learning, and my role is to help them find and hone their style. However, I made sure that the class was aware of end goals (learning objectives). In addition, I communicated my expectations very clearly to the class. Therefore, in the beginning of every class I would tell the students that what they would be able to do after learning the topic, that I was going to teach. Clearly communicating my expectations helped my students better understand the end goals (goal and development orient leadership).

I also like to own the outcome of my decisions as a leader. There were many instances where my expectations from the class did not match the reality. I, therefore, critically examined my mistakes and revised them. In addition, there were many instances during my PhD where some proof ideas did not go the way I expected but I learned from them by carefully examining my assumptions. I recently realised that communication is one of the key factors in effective leadership, but more importantly staying calm during a difficult conversation is one the most underrated skill.