Research Statement

Mukesh Tiwari

Fair elections are the only way to keep a democracy alive. In an election, most countries (election commission) use paper ballots to record its citizens' votes (preferences). Many countries, however, in recent years are introducing computers to conduct some part, or all, of election processes because it is cost effective, accessible to disabled voters, faster result, convenient, etc. In addition, some voting methods are very complex to count by hand. For example, single transferable vote used in Australian senate elections, and it may take months to declare the final result of a senate election if counted by hand. Therefore, the Australian Election Commission scans all the ballots of the senate election and uses a (closed source) software program to produce the final tally. Nonetheless, there is a growing debate about using electronic voting machines (computers) because of software bugs leading to unintended consequences, e.g., Rina Mercuri, a candidate in Australia, lost an election because of a software bug[1], SwissPost e-voting source code contained a serious bug[2], etc. Therefore, it is more imperative than ever that we mathematically prove correct (formally verify) the software programs used in elections. It will make the elections more trustworthy and establish the trust of general members of public in electronic voting.

1 PhD 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, correctness, privacy, and (universal) verifiability, of a paper ballot election to an electronic setting (electronic voting). In a paper ballot election, correctness and verifiability are ensured by scrutineers, while privacy comes for free because of secret paper ballots. However, achieving these three desirable properties are difficult in electronic voting because software programs used in various stages of an (electronic) election work in a opaque (blackbox) manner. This opaqueness was one of the major reason for Bundesverfassungsgericht, the German Constitutional Court, to rule the usage of electronic voting machines unconstitutional. However, it did not completely rule out the usage of electronic voting machines as long as the outcome of an election is verifiable, i.e., it is possible for the citizen to check the essential steps in the election act and in the ascertainment of the results reliably and without special expert knowledge[3] (verifiability).

In my thesis, I demonstrated the correctness of a vote-counting software program by implementing and proving the correctness of the Schulze method [1] 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

[1] https://www.areanews.com.au/story/3971893/mercuri-robbed/
[2] https://bit.ly/2TDlQOj
[3] https://www.bundesverfassungsgericht.de/SharedDocs/Entscheidungen/EN/2009/03/cs20090303_2bvc000307en.html

projects and political groups[4]. From my Coq implementation of the Schulze method, I used Coq's extraction mechanism to get an OCaml program, and I, then, used OCaml compiler to compile the extracted OCaml program to get an executable to count ballots [1]. In addition to correctness, my implementation also ensured (universal) verifiability by producing a scrutiny sheet. The scrutiny sheet contained all the data to audit the election independently. The (extracted) OCaml program, however, was very slow and could not count more than 10,000 ballots. Therefore, I wrote another Coq implementation [2], proven equivalent to the slow Coq implementation [1], from which the (extracted) OCaml program was able to count millions of ballots.

In both, slow and fast, Coq implementations, 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 [3, 4]. 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 publicly, on a bulletin board. The attack is: a coercer demands a voter to mark them as first and for the rest of candidates in a certain given order (permutation). Later, the coercer checks if that the order (permutation) appears on the bulletin board or not. Although my previous two Coq implementations [1, 2] satisfied correctness and verifiability criteria, they lacked privacy due to "Italian" attack. In order to avoid this attack on the Schulze method, I used a homomorphic encryption to encrypt the ballots and computed the winner by combining all the (encrypted) ballots, without decrypting any individual ballot (privacy). Moreover, I addressed verifiability by generating a scrutiny sheet (certificate) augmented with zero-knowledge-proofs for various claims, e.g., honest decryption, honest shuffle [5]. This work was carried out in the Coq theorem prover, so I ended up achieving correctness, privacy, and verifiability. The downside of an encrypted ballot Schulze election, or in fact any encrypted ballot election, is difficulty in auditing because of involved mathematics of cryptography. Therefore, in the future I want to explore the possibility of a verified prototype of scrutiny-sheet checker for encrypted ballots Schulze elections. Finally, I worked on a verified prototype of a simple approval voting election scrutiny-sheet checker for International Association of Cryptologic Research (IACR) [6]. In addition, I was involved in the formalisation of single transferable vote, used in the Australian Senate [7].

2 Future Work

My long-term aim is to make formal verification ubiquitous in software development, specifically for the software programs deployed in public domain that affect common people. My expertise in **Theorem Proving, Cryptography, Election Security,** and **Social Choice Theory** gives me an unique perspective to solve challenging problems that matter to many democracies and its citizens.

2.1 Mathematically Proven Correct Cryptographic Algorithms

Cryptographic algorithms are used ubiquitously to secure the data and correctness is an utmost requirement for any cryptographic algorithm implementation. Therefore, I will focus on developing mathematically proven correct cryptographic algorithms used in electronic voting, blockchain,

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

and secure communication, e.g., sigma protocols (zero-knowledge-proof), verifiable (shuffling) mix-networks, multi-party computations, secret-sharing, zkSNARK, etc. The rationale behind implementing these algorithms is that anyone can use them to construct an utility, e.g., an election scrutiny-sheet checker, a vote-tallying system based on blockchain, a verifiable ballot mixing service, an auction server, etc. One of the motivation behind this project is to replace the SwissPost Java implementations[5] with mathematically proven correct Coq implementations[6].

2.2 Mathematically Proven Correct Vote-Counting Algorithms

In future, I will focus on developing mathematically proven correct software programs for vote-counting methods used across the world such as *Single Transferable Vote (STV), First Past the Post (FPTP), Instant-runoff voting(IRV), etc.,* in the Coq theorem prover. All the vote-counting methods, by design, lend themselves well to computing the winner from plaintext ballots; however, so far there is very little research in computing the winner from encrypted ballots while ensuring correctness, privacy, and verifiability. Therefore, producing the winner from encrypted ballots is a challenging task. The motivation for this project is that once we have mathematically proven correct components, anyone –election commission or members of general public– can use them to conduct elections, referendums, and verify elections' outcome without worrying about software bugs. The cryptographic algorithms formalised in the previous step are going to be used as a building block in this project.

2.3 Mathematically Proven Correct Social Choice Properties

Computational social choice theory is a research area that is concerned with aggregation of ballots (preferences) of multiple voters (agents) and encompasses computer science, mathematics, economics, and political science. Typical applications of computational social choice theory is voting (preference aggregation), resource allocation, and fair division. Most of the proofs in computational social choice theory are pen-and-paper proofs, and one of my long term future research goal is to make them more precise using the Coq theorem prover. My current focus is voting because voting methods admit many excellent (social choice) properties established by political scientists, social choice theorists, and economists. For example, the Schulze method follows Condorcet criterion, reversal symmetry, polynomial runtime, etc., so when we formalise the Schulze method, or in fact any vote-counting method, we can push the boundary of correctness by proving that our implementation of the Schulze method also follows all the properties [8]. In addition, we can analyse these voting methods from computational complexity perspective of bribery, if by bribing a certain amount voters a specified candidate can be made an election's winner [9]. This research opens the door of collaboration with political scientists, social choice theorists, economists, game theorists, and theoretical computer scientists.

2.4 Mathematically Proven Correct Decentralised Application

I will focus on a mathematically proven correct decentralised peer-to-peer technical solution [10, 11, 12, 13] in the Coq theorem prover. The motivation is to help whistleblowers in leaking documents

[5] `https://bit.ly/3EODmnF`
[6] An ongoing project `https://github.com/mukeshtiwari/Dlog-zkp/`

and exposing corruption without revealing their identities. Being vocal against the government is one the most fundamental right of any citizen, but many authoritative governments do not appreciate dissent of any form. Therefore, it uses its powerful machinery to punish dissidents, in the name of national security. The inspiration for this project comes from David McBridge[7] and Richard Boyle[8]. David McBride is facing a threat of lifetime jail after leaking the material alleging war crimes by members of the Australia's Special Operations Task Group in Afghanistan, while Richard Boyle is facing 161 years for exposing the corruption inside the Australian Taxation office (Australia is ranked very high in democracy index[9]). This research will open the door of collaboration with many groups working in verified networking, and verified distributed systems.

3 Current Work at Cambridge and Previous Work at Melbourne

In my current project *Combinators for Algebraic Structures (CAS)*, I am formalising various graph algorithms on *semiring* algebraic structure and combinators (functions) to combine two, or more, algebraic structures. In this work, I am developing a mathematical correct-by-construction [1] framework, in Coq thereom prover, based on theory of generalised path-finding algebra [14, 15]. In our framework, depending on concrete instantiation of semiring operators, the same algorithm can compute shortest path, longest paths, widest paths, multi-objective optimisation, data flow of imperative programs, and many more [16]. In fact, the Schulze method is one instance of our framework. However, the current CAS implementation is highly focused towards networking protocols, so as a future work I will focus on adding more algorithms in CAS related to multi-objective optimisation, data flow analysis of imperative programs, and voting.

At the university of Melbourne, my work was focussed on constant-time implementations, a key requirement for many applications including cryptography. In particular, I worked on security concurrent separation logic for formally reasoning about the information flow properties of a concurrent program. I used SecureC, a tool developed at the university of Melbourne, to formalise an email server, an auction server, and a location server. In addition, I developed a information flow secure gradient descent algorithm (a machine learning algorithm) in SecureC for trusted execution environment, e.g., Intel SGX and ARM TrustZone to process highly sensitive data such as parents' income, race, gender, incarceration time, etc[10]. This work has been informally presented at PaveTrust workshop[11]. All these works were mathematically proven to leak no sensitive information to an attacker observing the execution of a program processing some secret data.

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

[7] `https://en.wikipedia.org/wiki/David_McBride_(whistleblower)`
[8] `https://bit.ly/3OQ6kbC`
[9] `https://worldpopulationreview.com/country-rankings/democracy-countries`
[10] https://opportunityinsights.org/
[11] `https://bit.ly/3XJxhBv`

References

[1] 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.

[2] 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 *International Joint Conference on Electronic Voting*, pages 66–83. Springer, 2017.

[3] J. Otten. Fuller disclosure than intended. 2003. Accessed on October 17, 2019.

[4] Josh Benaloh, Tal Moran, Lee Naish, Kim Ramchen, and Vanessa Teague. Shuffle-sum: coercion-resistant verifiable tallying for STV voting. *IEEE Trans. Information Forensics and Security*, 4(4):685–698, 2009.

[5] Thomas Haines, Dirk Pattinson, and Mukesh Tiwari. Verifiable Homomorphic Tallying for the Schulze Vote Counting Scheme. In Supratik Chakraborty and Jorge A. Navas, editors, *Verified Software. Theories, Tools, and Experiments*, pages 36–53, Cham, 2020. Springer International Publishing.

[6] 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.

[7] 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 Serdült, and David Duenas-Cid, editors, *Electronic Voting*, pages 51–66, Cham, 2018. Springer International Publishing.

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

[9] Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra. The complexity of bribery in elections. In *AAAI*, volume 6, pages 641–646, 2006.

[10] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In *Australasian Conference on Information Security and Privacy*, pages 325–335. Springer, 2004.

[11] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. *Freenet: A Distributed Anonymous Information Storage and Retrieval System*, pages 46–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[12] zzz (Pseudonym) and Lars Schimmer. Peer Profiling and Selection in the I2P Anonymous Network. In *PetCon 2009.1*, pages 59–70, 2009.

[13] Henry Corrigan-Gibbs and Bryan Ford. Dissent: Accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, page 340–350, New York, NY, USA, 2010. Association for Computing Machinery.

[14] R. C. Backhouse and B. A. Carré. Regular Algebra Applied to Path-finding Problems. *IMA Journal of Applied Mathematics*, 15(2):161–186, 04 1975.

[15] Timothy G. Griffin and Joäo Luís Sobrinho. Metarouting. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, page 1–12, New York, NY, USA, 2005. Association for Computing Machinery.

[16] Michel Gondran and Michel Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms*, volume 41. Springer Science & Business Media, 2008.