# Marie Curie Fellowship Proposal

Mukesh Tiwari

## 1 Objective:

In this project, our goal is to develop a mathematically proven correct and secure software (computer program) used in Switzerland's electronic (internet) voting, employed for elections and referendums, using formal methods.

## 2 Background and Problem:

Switzerland is one of the oldest democracy, and its citizens actively participate in regular decision making, four times in a year, and express their opinion by means of voting. In general, paper ballots are used by Swiss citizens to express their opinion; however, in recent years there is a growing debate in favour of electronic voting to address various problems, such as including more of its overseas voters, being more inclusive to towards visually impaired voters[1], minimising the number of informal paper ballots, and streamlining the vote counting process [1, 2]. In fact, electronic voting is already in practice since 2000 in Switzerland, and the first trial of electronic voting to a binding referendum was conducted in Geneva in 2003, followed by Neuchâtel and Zurich in 2005 [3]. Geneva developed its own software for electronic voting, meanwhile Zurich took the help of a private company Unisys [4] and Neuchâtel developed with the help of their own IT department in collaboration with another private company Scytl [5]. Other cantons in Switzerland used the software developed by Geneva, Zurich, and Neuchâtel to conduct their elections and referendums. Nonetheless, currently none of the cantons are developing their own software citing cost[2] and security, and they are using the software develop by SwissPost, with the help of Scytl, who entered into the competition in 2016 [6]. SwissPost and Scytl kept the source code of their software closed until 2019, but in 2019 they hosted it on GitLab[3]. It only took few months for researchers to break its cryptographic code, the most important part of their software [7, 8]. Consequently, Swiss Federal Chancellery stopped the cantons from using SwissPost software for 2020 elections and formed a committee of experts from various areas including electronic-voting, cryptography, computer network, and political science to analyse the current voting situation and asked for their recommendations[4]. The participating experts have produced their findings and one of the crucial recommendation was to develop the source code of an electronic voting software using formal method (mathematically prove correct code)[5]. Thus, our proposal for a formally verified software (computer program) for electronic voting in Switzerland.

[1] `https://www.usenix.org/conference/enigma2021/presentation/folini`

[2] `https://www.swissinfo.ch/eng/politics/digital-voting_geneva-shelves-e-voting-platform-on-cost-grounds/44577490`

[3] `https://gitlab.com/swisspost-evoting/`

[4] `https://www.bk.admin.ch/bk/en/home/politische-rechte/e-voting/berichte-und-studien.html` (*List of mandated experts.pdf*)

[5] `https://www.bk.admin.ch/bk/en/home/politische-rechte/e-voting/berichte-und-studien.html` (see *Summary of the Expert Dialog 2020.pdf*)

## 3 Methodology and Solution

It is well-understood in computer science community that software have bugs, and the most widely used method to eliminate bugs from a software is testing. Unfortunately, testing cannot eliminate all the bugs, and therefore, mathematicians and computer scientists have put many decades of research to develop a bug-free software using formal methods. In this project, we are going to use formal methods to develop a mathematically proven correct software using the Coq theorem prover [9], EasyCrypt [10], and Jasmin [11].

In the first phase of this project, our goal is to understand the specification of crypto-primitives[6], and implement and prove the crypto-primitives correct in the Coq, EasyCrypt, and Jasmin theorem prover. From our implementation, we intend to extract a mathematically proven correct software for the frontend that runs in the browser interacting with a voter and the backend that runs on the server of an election. Our goal is to extract both, frontend and backend, from the same formalisation. Currently, SwissPost has written the same algorithms in two languages: (i) Java for the backend, and (ii) TypeScript for the frontend. In principle, writing the same algorithm in two different languages may appear innocuous, but in practice it is problematic because they are, in general, written by two different teams and there is a possibility of two code bases diverging from each other on some corner cases, i.e., outputting two different values for the same input. On the other hand, we are using just one formalism to extract two code bases, divergence is not applicable for our case. The Coq theorem prover has code generator that produces web assembly code (frontend) from a Coq formalisation, but it does not produce a provable correct assembly code (backend). On the other hand, EasyCrypt and Jasmin produce a provable correct assembly code (backend) from its formalisation, but it lacks web assembly code (frontend) and therefore extending it to extract web assembly code will also be a part of this phase. We expect the duration of this phase to be 8 months.

In the second phase of this project, our goal is to understand the specification of the election verifier[7] and implement it in the Coq, EasyCrypt, and Jasmin theorem prover. From our implementation, we intend to extract a mathematically proven correct software that takes the cryptographic evidence of an elections and certifies if every step has been performed correctly or not. The rationale to produce another verifier is two fold:

- the current verifier[8] is written in Java and the only way to find a bug in Java program is to write test cases. However, as we eluded above, it is not possible to catch every single bug by means of testing, and therefore, it is possible that Java verifier may return incorrect result for the cryptographic evidence of an elections. On the other hand, our verifier is mathematically proven correct and hence it will never return incorrect result. In addition, our verifier can be used to detect bugs in Java verifier by feeding a random input to our verifier and comparing the output of our verifier to the Java verifier's output for the same random input.

- having an independent verifier will boost the confidence of voters in the system and reduce the barrier of electronic voting acceptance because the verifier will facilitate an additional independent way to verify an election's outcome [12]. Our verifier can be used by voters, election scrutineers, political

[6] https://gitlab.com/swisspost-evoting/crypto-primitives/crypto-primitives/-/blob/master/Crypto-Primitives-Specification.pdf

[7] https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/System/Verifier_Specification.pdf

[8] https://gitlab.com/swisspost-evoting/verifier/

parties to ensure the integrity of an election. Cryptographic primitives developed in the first phase will be used in this phase. We expect the duration of this (second) phase to be 8 months.

In the third phase of this project, our goal is to understand the whole e-voting system architecture[9] and how various components, e.g., voting client, voting server, return codes, etc., interact with each other. We would like to model this interaction mathematically using session types [13]. Session types are a formalism for distributed communicating processes, i.e., they encode various piece of information in a type system to mathematically reason about distributed communicating processes. It has found applications in numerous domains, e.g, distributed computing [14], multi-threaded functional programming [15], component-based systems [16], web programming [17], etc. However, Coq and EasyCrypt, in their current form, are not capable of reasoning about session types. Therefore, our first step will be extend these tools to reason about session types [18] and then use them to reason about the interaction between various components. This phase will be in two parts:

- the first part will be to develop a formal abstract model of interaction. Given that session types make the structure of communication explicit, we will get a formal model of interaction for the e-voting system architecture that can be used to reason about various properties, e.g., communication safety [19], secure information flow [20], computational complexity [21], etc.
- the second, and final, part will be to develop an implementation that respects the abstract model, produced in the first part. In addition, we can also check if the current Java implementation follows our abstract model or not, which will be a valuable insight about the current Java implementation. In this way, we close all the gaps by producing end-to-end verifiable system [22, 18, 23]. We expect the duration of the third phase to be 8 months.

# 4 Deliverables:

This project, if funded, will produce the following:

- output of the first phase: a mathematically proven correct software for the frontend and backend for cryptographic primitives (project duration: 8 months).
- output of the second phase: a mathematically proven correct software that takes the cryptographic evidence of an elections and certifies if every step has been performed correctly or not (project duration: 8 months).
- output of the third phase: a mathematical model of interaction and its implementation for the e-voting system architecture (project duration: 8 months).

Our mathematically proven correct software written in Coq, EasyCrypt, and Jasmin can be used to test the current Java code written by SwissPost. In addition, we will significantly improve the understanding of interaction by producing a formal model in the third phase of this project. This project, if funded, will not only produce the mathematically proven correct software but it will also enhance our understanding the whole system that include interaction amongst various components and the security of voting protocol.

In addition, our project will bring more transparency because not only we will open source the software but we will also publish papers in security conferences. Our claims will go through a rigorous review process

[9] `https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation`

of the program committee –leading security experts– of security conferences, a very different situation from software companies' practices. Companies providing software for electronic voting always cite intellectual property to not release the source and create a very opaque situation. This opaqueness was one of the major reason for Bundesverfassungsgericht[10], the German Constitutional Court, to rule the usage of electronic voting machines unconstitutional. It, however, 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 citizens to check the essential steps and ascertain the results reliably and without any special expert knowledge (verifiability).

## References


[1] Uwe Serdult, Micha Germann, Fernando Mendez, Alicia Portenier, and Christoph Wellig. Fifteen years of internet voting in Switzerland [History, Governance and Use]. In *2015 Second International Conference on eDemocracy and eGovernment (ICEDEG)*, pages 126–132, 2015.

[2] Ardita Driza-Maurer, Oliver Spycher, Geo Taglioni, and Anina Weber. E-voting for Swiss Abroad: A Joint Project between the Confederation and the Cantons. In Manuel J. Kripp, Melanie Volkamer, and Rüdiger Grimm, editors, *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, volume P-205 of *LNI*, pages 173–187. GI, 2012.

[3] Jan Gerlach and Urs Gasser. Three case studies from switzerland: E-voting. *Berkman Center Research Publication No*, 3:2009, 2009.

[4] Giampiero E.G. Beroggi. Secure and Easy Internet Voting. *Computer*, 41(2):52–56, 2008.

[5] Markus Reiners. Vote electronique in Switzerland: Comparison of relevant pilot projects. *Journal of Comparative Politics*, 13(1):58–75, 2020.

[6] Thomas Haines, Olivier Pereira, and Vanessa Teague. Running the Race: A Swiss Voting Story. In Robert Krimmer, Melanie Volkamer, David Duenas-Cid, Peter Rønne, and Micha Germann, editors, *Electronic Voting*, pages 53–69, Cham, 2022. Springer International Publishing.

[7] Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 644–660, 2020.

[8] Philipp Locher, Rolf Haenni, and Reto E Koenig. Analysis of the cryptographic implementation of the swiss post voting protocol, 2019.

[9] The Coq Development Team. The Coq Proof Assistant, version 8.10.0, October 2019.

[10] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-Aided Security Proofs for the Working Cryptographer. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 71–90, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[11] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Arthur Blot, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Hugo Pacheco, Benedikt Schmidt, and Pierre-Yves Strub. jasmin: High-assurance and high-speed cryptography.


[10] `https://www.bundesverfassungsgericht.de/SharedDocs/Entscheidungen/EN/2009/03/cs20090303_2bvc000307en.html`

[12] Ronald L Rivest. On the notion of 'software independence'in voting systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3759–3767, 2008.

[13] Laura Bocchi, Tzu-Chun Chen, Romain Demangeon, Kohei Honda, and Nobuko Yoshida. Monitoring Networks through Multiparty Session Types. In Dirk Beyer and Michele Boreale, editors, *Formal Techniques for Distributed Systems*, pages 50–65, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[14] Simon Fowler, Sam Lindley, J. Garrett Morris, and Sára Decova. Exceptional Asynchronous Session Types: Session Types without Tiers. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019.

[15] Vasco T Vasconcelos, Simon J Gay, and António Ravara. Type checking a multithreaded functional language with session types. *Theoretical Computer Science*, 368(1-2):64–87, 2006.

[16] Antonio Vallecillo, Vasco T Vasconcelos, and António Ravara. Typing the behavior of objects and components using session types. *Electronic Notes in Theoretical Computer Science*, 68(3):439–456, 2003.

[17] Anson Miu, Francisco Ferreira, Nobuko Yoshida, and Fangyi Zhou. Communication-safe web programming in typescript with routed multiparty session types. In *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*, CC 2021, pages 94–106, New York, NY, USA, 2021. Association for Computing Machinery.

[18] Fangyi Zhou, Francisco Ferreira, Raymond Hu, Rumyana Neykova, and Nobuko Yoshida. Statically Verified Refinements for Multiparty Protocols. In *OOPSLA 2020: Conference on Object-Oriented Programming Systems, Languages and Applications*, volume 4 of *PACMPL*, pages 148:1–148:30. ACM, 2020.

[19] Bernardo Toninho and Nobuko Yoshida. Certifying data in multiparty session types. *Journal of Logical and Algebraic Methods in Programming*, 90:61–83, 2017.

[20] Sara Capecchi, Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Tamara Rezk. Session Types for Access and Information Flow Control. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory*, pages 237–252, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[21] Romain Demangeon and Nobuko Yoshida. Causal computational complexity of distributed processes. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 344–353, New York, NY, USA, 2018. Association for Computing Machinery.

[22] Ricardo Corin, Pierre-Malo Denielou, Cedric Fournet, Karthikeyan Bhargavan, and James Leifer. Secure Implementations for Typed Session Abstractions. In *20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 170–186, 2007.

[23] L. Arquint, F. A. Wolf, J. Lallemand, R. Sasse, C. Sprenger, S. N. Wiesner, D. Basin, and P. Müller. Sound Verification of Security Protocols: From Design to Interoperable Implementations. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1077–1093, Los Alamitos, CA, USA, may 2023. IEEE Computer Society.