Verifiable Homomorphic Tallying for the Schulze Vote Counting Scheme

Mukesh Tiwari
mukesh.tiwari@anu.edu.au
Research School of Computer Science,
Australian National University
Canberra

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

Thomas Haines
thomas.haines@ntnu.no
Dept of Mathematical Sciences,
Norwegian University of Science and
Technology, Trondheim, Norway

ABSTRACT

The encryption of ballots is crucial to maintaining integrity and anonymity in electronic voting schemes. It enables, amongst other things, each voter to verify that their encrypted ballot has been recorded as cast, by checking their ballot against a bulletin board.

We present a verifiable homomorphic tallying scheme for the Schulze method that allows verification of the correctness of the count—on the basis of encrypted ballots—that only reveals the final tally. We achieve verifiability by using zero knowledge proofs for ballot validity and honest decryption of the final tally. Our formalisation takes places inside the Coq theorem prover and is based on an axiomatisation of cryptogtaphic primitives, and our main result is the correctness of homomorphic tallying. We then instantiate these primitives using an external library and show the feasibility of our approach by means of case studies.

1 INTRODUCTION

Since the introduction of secret ballot by Australians in 1855, paper (ballots) are widely used around the world to record the preferences of eligible voters. Paper ballots provide three important ingredient, correctness, privacy, and verifiability, for any democratic election. However, the paper ballot election poses various other challenges, e.g. slow for large democracies like India, error prone for complex voting method like single transferable vote, and poses operational challenges for massive countries like Australia. In order to solve these problems and various others, many countries are adopting electronic voting. However, electronic voting has a whole new set of problems. In most cases, the software program used to conduct the election has numerous problems, including, but no limited to, counting bugs, ballot identification, etc. Moreover, these software programs are treated as commercial in confidence [1] and are not allowed to be inspected by general member of public. As a consequence, the result produced by these software programs can not be substantiated with verifiable evidence of their correctness while retaining the secrecy of the individual ballot [4]. Technically, the notion of "verifiable evidence" is captured by the term *end-to-end (E2E) verifiability*, that is

- Cast-as-intended: every voter can verify that their ballot was cast as intended.
- Collected-as-cast: every voter can verify that their ballot was collected as cast
- Tallied-as-cast: everyone can verify final result on the basis of the collected ballots.

The combination of privacy and verifiability can be realised using cryptographic techniques, where encrypted ballots (that the voters themselves cannot decrypt) are published on a bulletin board, and the votes are then processed, and the correctness of the final tally is substantiated, using homomorphic encryption [7] and verifiable shuffling [2]. Verifiability can then be guaranteed by means of Zero Knowledge Proofs (ZKP), first studied by Goldwasser, Micali, and Rackoff [6].

2 SCHULZE METHOD

The Schulze Method [8] is a vote counting scheme that elects a single winner, based on preferential votes. The method can be described as:

- Consider an election with a set of m candidates $C = \{c1, \ldots, cm\}$, and a set of n votes $P = \{b1, \ldots, bn\}$. A vote is represented as function $b : C \longrightarrow \mathbb{N}$ that assigns natural number (the preference) to each candidate. We recover a strict linear preorder $<_b$ on candidates by setting $c <_b d$ if $b(c) > b(d)$.
- Given a set of ballots P and candidate set C, we construct graph G based on the margin function $m : C \times C \longrightarrow \mathbb{Z}$. Given two candidates $c, d \in C$, the *margin* of c over d is the number of voters that prefer c over d, minus the number of voters that prefer d over c. In symbols:

  $$m(c, d) = \sharp\{b \in P \mid c >_b d\} - \sharp\{b \in P \mid d >_b c\}$$

  where $\sharp$ denotes cardinality and $>_b$ is the strict (preference) ordering given by the ballot $b \in P$.
- A directed *path* in the graph, G, from candidate c to candidate d is a sequence $p \equiv c_0, \ldots, c_{n+1}$ of candidates with $c_0 = c$ and $c_{n+1} = d$ ($n \geq 0$), and the *strength*, st, of path, p, is the minimum margin of adjacent nodes, i.e.

  $$st(c_0, \ldots, c_{n+1}) = \min\{m(c_i, c_{i+1}) \mid 0 \leq i \leq n\}.$$

- For candidates c and d, let $M(c, d)$ denote the maximum strength, or generalized margin of a path from c to$d$ i.e.

  $M(c, d) = \max\{st(p) : \text{p is path from c to d in G}\}$
- The winning set is defined as

  $W = \{c \in C : \forall d \in C \setminus \{c\}, M(c, d) \geq M(d, c)\}$

Now we describe the homomorphic Schulze method in short. We first homomorphically compute the margin matrix from encrypted ballots, and then compute winners on the basis of the (decrypted) margin. The computation also produces a verifiable certificate that leaks no information about individual ballots other than the (final) margin matrix, which in turn leaks no information about individual ballots if the number of voters is large enough.

Format of Ballots. In preferential voting schemes, ballots are rank-ordered lists of candidates. For the Schulze Method, we require that all candidates are ranked, and two candidates may be given the same rank. That is, a ballot is most naturally represented as a function $b : C \longrightarrow \mathbb{N}$ that assigns a numerical rank to each candidate, and the computation of the margin amounts to computing the sum

$$m(x, y) = \sum_{b \in B} \begin{cases} +1 & b(x) > b(y) \\ 0 & b(x) = b(y) \\ -1 & b(x) < b(y) \end{cases}$$

where B is the multi-set of ballots, and each $b \in B$ is a ranking function $b : C \longrightarrow \mathbb{N}$ over a (finite) set C of candidates.

We note that this representation of ballots is not well suited for homomorphic computation of the margin matrix as practically feasible homomorphic encryption schemes do not support comparison operators and case distinctions as used in the formula above.

We instead represent ballots as matrices $b(x, y)$ where $b(x, y) = +1$ if x is preferred over y, $b(x, y) = -1$ if y is preferred over x and $b(x, y) = 0$ if x and y are equally preferred.

While the advantage of the first representation is that each ranking function is necessarily a valid ranking, the advantage of the matrix representation is that the computation of the margin matrix is simple, that is

$$m(c, d) = \sum_{b \in B} b(x, y)$$

where B is the multi-set of ballots (in matrix form), and can moreover be transferred to the encrypted setting in a straight forward way: if ballots are matrices $e(x, y)$ where $e(x, y)$ is the encryption of an integer in $\{-1, 0, 1\}$, then

$$= \bigoplus_{\in} (x, y) \quad (1)$$

where $\oplus$ denotes homomorphic addition, is an encrypted ballot in matrix form (i.e. decrypting (x, y) indicates whether x is preferred over y), and is the multi-set of encrypted ballots. The disadvantage is that we need to verify that a matrix ballot is indeed valid, that is

- that the decryption of (x, y) is indeed one of $1, 0$ or -1
- that indeed corresponds to a ranking function.

Indeed, to achieve verifiability, we not only need *verify* that a ballot is valid, we also need to *evidence* its validity (or otherwise) in the certificate. In order to decide the validity of any ballot, b, we first generate a random permutation matrix, π, and commit each column of π using the Pedersen's commitment scheme producing a list of commitments, c_π (hiding phase). Moreover, we keep the π secret and publish the c_π (make it public). Now we shuffle each row of b by π producing a row-shuffled ballot, $b_{r\pi}$, and a shuffle zero-knowledge proof, $b_{zkpr\pi}$. Subsequently, every column of the ballot $b_{r\pi}$ is shuffled by the same (secret) permutation π producing a column-shuffled ballot, $b_{rc\pi}$, and a shuffle zero-knowledge proof, $b_{zkprc\pi}$. Finally, we decrypt $b_{rc\pi}$ producing a decrypted ballot, $decb_{rc\pi}$, and zero-knowledge proof of honest decryption, $decb_{zkprc\pi}$. Finally, we publish the whole data (except the secret permutation π) on the bulletin board (scrutiny sheet). Couple of important points to be noted:

- shuffle introduces re-encryption, so π can not be guess from the data b and $b_{r\pi}$, or from the commitment c_π
- every claim is augmented with a corresponding zero-knowledge-proof to make it verifiable
- for the binding step in Pedersen's commitment, rather than opening π we generate a zero knowledge proof, zkp_π, using π and c_π which can be used to prove that c_π is indeed the commitment to some permutation which has been used in the (commitment consistent) shuffling without being opened [9].

We have shown that Schulze method can be implemented in a theorem prover to guarantee both provably correct and verifiable counting on the basis of encrypted ballots, relative to an axiomatisation of the cryptographic primitives. We then obtain, via program extraction, a provably correct implementation of vote counting, that we turn into executable code by providing implementations of the primitives based on a standard cryptographic library. Finally, in a nutshell, we achieve: (i) *correctness* by formally specifying the Schulze method and prove its correctness properties inside the Coq theorem prover, (ii) *privacy* by using homomorphic encryption to compute the final tally without decryption any individual ballot, and (iii) *verifiability* by tabulating the relevant data of election (scrutiny-sheet/certificate). We achieve verifiability in encrypted ballot counting by augmenting the scrutiny sheet with zero-knowledge-proof for the each claim we make during the counting, which can later be checked by any auditor.

REFERENCES

[1] Australian Electoral Commission. 2013. Letter to Mr Michael Cordover, LSS4883 Outcome of Internal Review of the Decision to Refuse your FOI Request No. LS4849. avaliable via http://www.aec.gov.au/information-access/foi/2014/files/ls4912-1.pdf, retrieved January 2, 2020.

[2] Stephanie Bayer and Jens Groth. 2012. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In *Proc. EUROCRYPT 2012 (Lecture Notes in Computer Science)*, David Pointcheval and Thomas Johansson (Eds.), Vol. 7237. Springer, 263–280.

[3] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. 1988. Everything Provable is Provable in Zero-Knowledge. In *CRYPTO (Lecture Notes in Computer Science)*, Vol. 403. Springer, 37–56.

[4] Matthew Bernhard, Josh Benaloh, J. Alex Halderman, Ronald L. Rivest, Peter Y. A. Ryan, Philip B. Stark, Vanessa Teague, Poorvi L. Vora, and Dan S. Wallach. 2017. Public Evidence from Secret Ballots. In *Proc. E-Vote-ID 2017 (Lecture Notes in Computer Science)*, Robert Krimmer, Melanie Volkamer, Nadja Braun Binder, Norbert Kersting, Olivier Pereira, and Carsten Schürmann (Eds.), Vol. 10615. Springer, 84–109.

[5] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1991. Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems. *J. ACM* 38, 3 (1991), 691–729.

[6] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1985. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In *STOC*. ACM, 291–304.

[7] Martin Hirt and Kazue Sako. 2000. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *Proc. EUROCRYPT 2000 (Lecture Notes in Computer Science)*, Bart Preneel (Ed.), Vol. 1807. Springer, 539–556.

[8] Markus Schulze. 2011. A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method. *Social Choice and Welfare* 36, 2 (2011), 267–303.

[9] Douglas Wikström. 2009. A Commitment-Consistent Proof of a Shuffle. In *Proceedings of the 14th Australasian Conference on Information Security and Privacy (ACISP '09)*. Springer-Verlag, Berlin, Heidelberg, 407–421. https://doi.org/10.1007/978-3-642-02620-1_28