Anonymous Electronic Voting System on Public Blockchains

Edilson Osorio Junior
osoriojr@gmail.com
13 January 2018
Version 1.0

Abstract

An electronic voting system, to be trusted, lies on being auditable and widely verifiable. To achieve an extraordinary level of transparency, all the ballots and votes must be public. However, extreme transparency could bring privacy issues, and because of that, the actual model uses a private bulletin board for registering, mixing, counting and publishing the result of the votes, and all participants must trust it. Some attempts of doing voting on blockchain use linked ring signatures for issuing the vote and for proving the voter already voted, unlinking it of the original identity of the voter. However, unfortunately, the use of ring signatures rely on a session of registration of all participants first and opening the voting session just after getting all public keys fo the voters. It is a time-consuming problem and a voter who does not find the moment of registration session open, could not be enabled to vote later. Our approach uses stealth addresses with zero-knowledge proof-of-vote, Paillier encryption for votes and Ethereum as a framework for public storage and smart-contracts to provide a fully auditable system, during and after the voting session.

Keywords: electronic voting, balloting, e-voting, blockchain, ethereum, privacy, zero-knowledge proof, homomorphic encryption

1. Introduction

Most voting processes rely on a trust in the agent who has contact with the votes, be it governmental or even in public and private companies. Because the vote goes to a ballot box (a container or black box) and after the vote is manipulated away from our eyes, we can not confirm if our vote has been altered or even counted. We could not even check if more votes were added, after or before, to the same ballot box.

One example of this hassle, involving a big public company in 2017, affected Procter&Gamble where a major proxy voting dispute got more than $60m until it gets resolved[1]. In that same year, the total sum of the expenses with proxy voting of four big companies took more than $140m. Among the main problems presented was the fact of the uncertainty surrounding the casted vote. The shareholders were unable to determine if their vote was counted in the election since it goes through many hands until reaching the destination. Because of that, there was a costly process of recounting and collapse of trust in the third-party agents.

Some initiatives attempt to solve the issue without the use of cryptography[2], as others try to create a secure environment using several layers of strong encryption at different times.

Because paper voting is easily frauded, many e-voting initiatives have appeared around the world. However, most of them are centralised and rely on trust in the agent who will have contact with the vote.

There are many challenges in designing the electronic voting architecture, especially regarding privacy and verifiability of the vote, as they are seemingly conflicting topics.

Recently many initiatives have tried to use blockchain to store votes and make use of other features inherent to the protocol, such as immutability, audibility, decentralisation and the possibility of running decentralised applications automatically and without human interference, among others.

[1] (Racanelli)
[2] (Rivest)

The protocol that we are presenting makes use of blockchain, and it is based on proof-of-knowledge[3][4][5][6], using many encryption algorithms to maintain the privacy and the trust in several layers and phases of the electoral process.

Unlike other projects, this process does not use linked ring signatures[7][8] because it demands a unique session of registration of all voters, who can start their voting process only after the conclusion of the registration process. In the case of an election with several voters, this session will remain open for many days or weeks and, after its completion, it would not be possible for a voter to be allowed to vote without weakening the trust in the process.

Thus, we constructed a protocol that premises the vote-and-go, which can be applied in a majority or preferential voting systems, that does not rely on a trusted third party agent to manipulate and tally votes or publish the result. This protocol can be used for any public blockchain that has the functionality of decentralised applications and has proven to be highly scalable.

2. Background

2.1 Zero-knowledge proof and zk-snarks

Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than correctness of the proposition in question[9]. Zk-snarks is an acronym to "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge" and refers to a proof construction where one can prove possession of specific information, e.g. a secret key, without revealing that information, and without any interaction between the prover and verifier[10].

2.2 Stealth voter wallet

Peter Todd presented the first protocol for stealth address, and it is a new type of Bitcoin address and related scriptPubKey/transaction generation scheme that allow payees to publish a single, fixed, address that payers can send funds efficiently, privately, reliably and

[3] (Camenisch et al.)
[4] (Damgård)
[5] (Shafi Goldwasser et al.)
[6] (Blum et al.)
[7] (Tsang and Wei)
[8] (Wu; Lai et al.)
[9] (S. Goldwasser et al.)
[10] (*What Are Zk-SNARKs?*)

non-interactively. Payers do not learn what other payments have been made to the stealth address, and third-parties learn nothing at all[11]. On this work we propose a derivation of this method for creating ethereum stealth addresses, that will be used for sending the ballot to the smart-contract.

2.3 Homomorphic Encryption

Fully homomorphic encryption, or merely homomorphic encryption, refers to a class of encryption methods envisioned by Rivest, Adleman, and Dertouzos already in 1978, and first constructed by Craig Gentry in 2009. Homomorphic encryption differs from typical encryption methods in that it allows computation to be performed directly on encrypted data without requiring access to a secret key. The result of such a computation remains in encrypted form, and can at a later point be revealed by the owner of the secret key[12].

The Homomorphic Encryption has a standard[13], and we chose the Paillier algorithm on our Proof-of-Concept ("PoC")
We chose to use the Paillier protocol[14] because there are available tools for encrypting/decrypting and create a zero-knowledge proof-of-result, aligned to the purposes of our protocol (Annex A)

2.4 Key Management

Actually, information about the summarising or counting the ballots and vote results should not be public before the tallying phase. To achieve that, we use Paillier encryption, because it enables Accounting the ballots and votes, without exposing the vote content. It uses a key pair where the private key is intended not to be published and is stored in a very secure environment. The public key must be released on a smart-contract, where it could be found by all voters, who use the key for encrypting their ballots during the casting of votes process.

2.5 Encrypted note

It is an abstraction of a message, encrypted on a private transaction that can be opened just by the ownership of the note, who has knowledge of the viewing key for opening the encrypted note. It could implement the concepts of encrypted notes provided by Zcash[15] or

[11] (*Bitcoin-Development | Stealth Addresses*)
[12] (*Introduction – Homomorphic Encryption Standardization*)
[13] (Albrecht et al.)
[14] (Dahlin and daylighting society)
[15] (*Zcash Protocol Specification*)

Aztec[16] protocol for confidential transactions on Ethereum. We would use the concept for encrypting the pubscan key for sending it to the Voter Registration Administrator, who provides the stealth voter addresses that are used for unlinking the voting address from the identity of the voter.

2.5 Identity validation

On any election, the identity of the voter must be validated and be present on a whitelist of voters. We use blockchain for storing the identities of the voters enabled to vote on a voting session as a whitelist format. A trusted third-party could be used for validating the identity of the users.

2.6 Blockchain

The blockchain provides Bitcoin's public ledger, an ordered and timestamped record of transactions[17]. This system is used to protect against double spending and modification of previous transaction records. Blockchain is being widely used for timestamping documents, for registering identities and making use of decentralised applications that are immutable, public, verifiable and auditable.

2.7 Ethereum

Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralised applications where they can create their own arbitrary rules for ownership, transaction formats, and state transition functions[18]. Smart contracts, cryptographic "boxes" that contain value and only unlock it if certain conditions are met, can also be built on top of the platform[19].

3. Related Work

Some cryptographic methods are being applied to general bulletin based voting systems and blockchain based voting systems[20]:

[16] (*Aztec Protocol Specification*)
[17] (*Developer Guide - Bitcoin | Blockchain*)
[18] (*A Next-Generation Smart Contract and Decentralized Application Platform*)
[19] (*Decentralised Applications*)
[20] (Yu et al.)

3.1 Public bulletin based voting system:

- Homomorphic encryption
- Mix-net
- Zero-knowledge proof
- Blind signature and linkable ring signature

3.2 Blockchain based voting systems:

- Voting systems using cryptocurrency
- Voting systems using smart-contract
- Voting systems using blockchain as a ballot box

4. Proposed Voting Protocol

4.1. Entities
4.2. Voting phases
4.2.1. Voting setup
4.2.2. User validation phase
- Validate identity
- Setup for enabling the user for voting

4.2.3. Voting phase
4.2.4. Tallying (counting) phase
4.2.5. Auditing phase

4.1 Entities

- *Identity validator*: an entity with the ability to validate the voter identity and to define if the potential voter can participate in this voting session;

- *Voter registration administrator*: ephemeral entity entitled for setting up the environment for user voting. Its role is mainly to create the stealth voter wallets who are used for voting, based on the encrypted note with a pubscan key sent by a voter;

- *Voting administrator*: the entity responsible for setting up the all voting environment. It registers the voting whitelist, prepares the trusted setup for zero-knowledge proofs and publishes the ballot encoding public key;

- *Smart-contract administrator*: working together with the voting administrator, is responsible for deploying the smart-contracts;

- *Smart-contracts*: one or more smart-contracts with the roles: 1) store the voting whitelist, 2) enable/disable voters, 3) store the stealth voter wallets, 4) store the encrypted votes, 5) validate and store the proof-of-vote, 6) verify the validity of the stored votes, 7) count the encrypted votes and voters, 8) publish the vote result, 9) verify the vote result;

- *Voters:* they are the people who have rights to vote on the enabled voting session. Voters get their identity validated and then, anonymously, cast their votes. After casting a vote, the voter can check the ledger to see the recorded vote.

4.2 Voting phases

The proposed protocol uses five different phases: setup, validating/registering voters, voting, tallying and auditing.

4.2.1 Voting setup phase

The voting administrator should set up the environment for the voting session:

- Setup the zero-knowledge proofs to enable users for creating their own (zk)proof-of-vote without publishing their votes;
- Create the homomorphic encryption keypair: is used for encrypting/decrypting votes. The public key must be published on the smart-contract for being available to voters who encrypt their votes;
- Provide the voting whitelist;
- Count votes after the ending of the voting session;
- Validate result and provide the proof-of-result.

The smart-contracts administrator should:

- Publish the verification smart-contract, for validating the proof-of-votes based on the zero-knowledge proof setup;
- Publish the voting session smart-contract(s) and enable it;
- Publish the voting whitelist;
- Publish the public key that voters use for encrypting their votes;
- Open and close the vote session;

- Publish the proof-of-result.

4.2.2 Voter validation phase

This phase involves two entities: identity validator and voter registration administrator. The identity validator is a neutral entity who validates the identity of the voter and enable her/him to vote. The voter registration administration has no contact with real voter identities. Receives the anonymous encrypted note to create and enable a stealth voter wallet, which is used for voting.

4.2.3 Voting phase

The voter conducts the voting phase:

- Must discover the enabled stealth voter wallet that can be used for voting;
- Cast their votes for all candidates in each race, filling the voting ballot (Annex B);
- Sign the ballot using its own validated identity;
- Recover the public key already published and encrypt each vote in the ballot;
- Create a proof-of-vote, for proving the vote without exposing sensitive information about it;
- Publish the encrypted ballot directly to blockchain using the stealth voter wallet;
- (optionally) Publish the proof-of-vote own validated identity.

4.2.4 Tallying phase

The voting administrator and smart-contracts administrator conduct the tallying phase:

- Smart-contract adds all ballots together and publishes the encrypted result;
- Voting administrator fetch the encrypted sum of all ballots, decrypt it and create the proof-of-result of the plain text result;
- Smart-contract administrator sends the proof-of-result to the smart-contract;
- Smart-contract verifies the proof-of-result and if true, stores the proof-of-result and the plain result.

4.2.5 Auditing phase

The auditing phase can be conducted by voters, voting administration and any trusted and neutral auditor.

All people: can check the total of voters and the votes.

Voters: can verify if own vote is stored on blockchain and can check the total of voters and votes.

Voting administrator and any trusted neutral external auditor: can use the private key for decrypting all votes, the encrypted result and check if both results are equal.

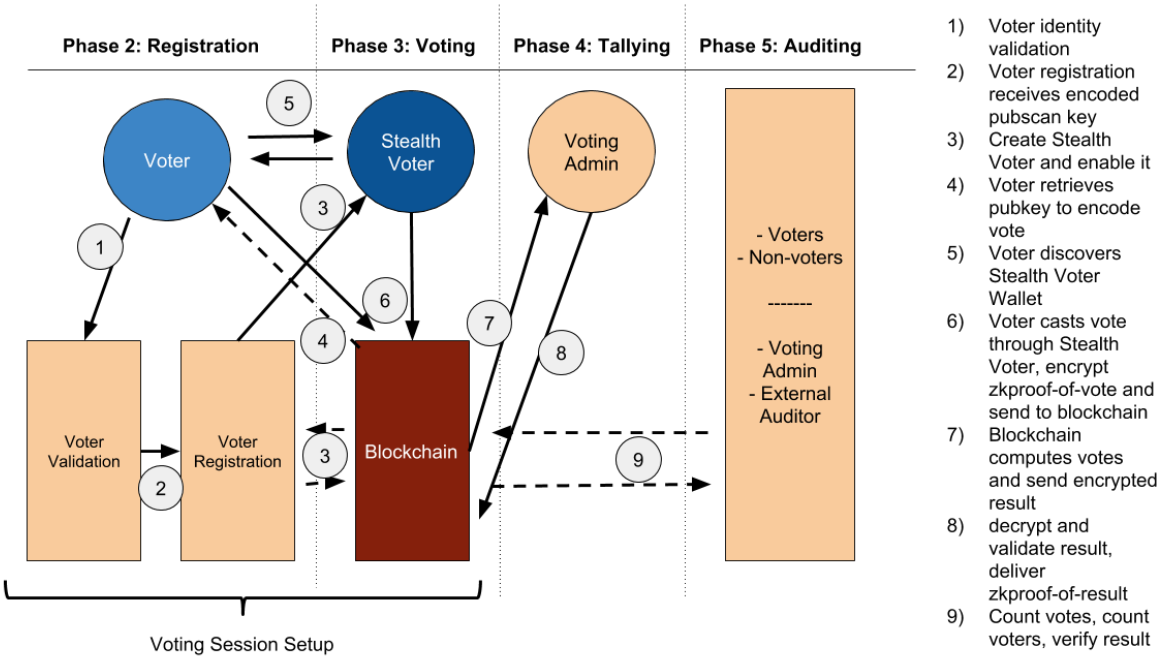

Fig 1. Voting Session Protocol

5. Security Analysis

5.1 Privacy

All the ballots and votes on the blockchain are encrypted. Just the Voting Administrator who set up the voting session can decrypt votes. No-one can know the result before the ending of the voting session and before voting administrator publishing the result and proof-of-result.

5.2 Anonymity

It is not possible to link the stealth voter address to the identity of the real voter, because of the use of stealth addresses.

5.3 No double-voting

Because of the use of a blockchain and smart-contracts, it is easy to detect if a voter tries to generate two or more votes. After generating a valid vote, the stealth voter address is marked as voted and cannot issue another vote. On systems that allow users to change their votes during the voting session, it is possible to track all new votes and account just the last one after ending the voting session.

5.4 Slanderability avoided

It is not possible to fake votes because of the model of signing messages using PKI and the transaction protocol of a public blockchain.

5.5 Receipt (proof-of-vote) available or receipt-freeness

This protocol enables the user to generate a zero-knowledge proof-of-vote when it is necessary to prove the voter voted and the vote is legit. As the voters cast their votes using the stealth vote address, it is not possible to prove what is its vote.

5.6 Public verifiability

Voters can verify their votes and if they are correctly registered. Non-voters can count votes and voters, and confirm if the calculation is correct.

5.7 Correctness

Zero-knowledge proofs ensure the completeness, knowledge soundness and a statistical zero-knowledge, which means it is possible to prove the correctness of the vote without exposing it, as the same for the proof-of-result.

5.8 Vote-and-go

This model presents a way for avoiding a phase where is needed to register all voters before starting the voting phase (because some systems use ring signatures schemes). Therefore, there is not required finishing the voting process to tally all votes because it can be possible, optionally, during the voting session.

5.9 Coercion resistance

This model makes it impossible to launch Ballots-buyer attack or Double voting attack. It is infeasible to an adversary to determine the vote by a coercive way. This model avoids attacks trying to nullify ballots because the voter must be validated for starting its voting session, and after receiving a valid vote, the voting session could be closed to the voter automatically by the smart-contract.

6. Additional considerations

6.1 Estonia e-voting

Estonia presented a very successful model for avoiding the coercion attempts[21]: they have two different sessions for voting, the first is electronic and the second happens in person. After closing the first session, it is opened the second session for voting in person, which means both sessions do not occur at the same time. People are allowed to change their votes during a window of time. Thus, if a person is coerced during the electronically voting session, it is allowed to go to the second moment, to cast a new vote in person. There is some criticism about many aspects of their e-voting system[22], but in reality, it is working for many years without much discussions about the security and privacy around their centralised model.

6.2 Ethereum gas consumption

There is a window for improvements on the gas consumption of the encrypted ballot registration, among other optimisations. Optionally, it could be stored on IPFS, and storing on-chain just the hash of the encrypted ballot.

7. Conclusion

The proposal of this protocol for an anonymous electronic voting system on public blockchains requires a minimal trust on the organiser. The transparency and auditability provided by a public blockchain like Ethereum bring another level of trust and security because everything can be auditable during the voting process. The smart-contract starts the tally phase and verify it using distributed computing if needed. The voting privacy is granted by stealth wallets, homomorphic encryption, at the same time that zero-knowledge proofs grant the and the proof-of-vote and the proof-of-result.

The proof-of-concept also suggests that our voting protocol is scalable and can perform even on large-scale voting.

[21] (Heiberg et al.)
[22] (Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, J. Alex Halderman)

8. Acknowledgements

Many people provided much valuable feedback. Among them, we would like to specially thanks to OriginalMy team, Miriam Oshiro, Danilo Salles a.k.a intrd (dann.com.br), Miao ZhiCheng and Jaanek Oja (Decentral.ee), Joao Ferreira (Brazilian Parliament), Fernando Ulrich (XDEX Exchange) and Nicolas Wagner (Kleros).

9. References

Albrecht, Martin, et al. *Homomorphic Encryption Standard*. 21 Nov. 2018, http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf.

A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 13 Jan. 2019.

Aztec Protocol Specification. https://github.com/AztecProtocol/AZTEC/blob/master/AZTEC.pdf. Accessed 13 Jan. 2019.

Baudron, Olivier, et al. "Practical Multi-Candidate Election System." *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing - PODC '01*, 2001, doi:10.1145/383962.384044.

Bitcoin-Development | Stealth Addresses. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html. Accessed 14 Jan. 2019.

Blum, Manuel, et al. "Noninteractive Zero-Knowledge." *SIAM Journal on Computing*, vol. 20, no. 6, 1991, pp. 1084–118.

Camenisch, Jan, et al. "Efficient Protocols for Set Membership and Range Proofs." *Lecture Notes in Computer Science*, 2008, pp. 234–52.

Dahlin, Taylor Fox, and daylighting society. "Paillier Zero-Knowledge Proof."

Https://paillier.daylightingsociety.org, 17 Dec. 2016, https://paillier.daylightingsociety.org/Paillier_Zero_Knowledge_Proof.pdf.

Damgård, Ivan. *On Σ-Protocols*. http://www-cs.ccny.cuny.edu/~fazio/F15-csc85030/readings/Dam10.pdf. Accessed 13 Jan. 2019.

Decentralised Applications. https://github.com/ethereum/wiki/wiki/Decentralized-apps-(dapps).

Developer Guide - Bitcoin | Blockchain. https://bitcoin.org/en/developer-guide#block-chain. Accessed 14 Jan. 2019.

Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, J. Alex Halderman. *Security Analysis of the Estonian Internet Voting System*. https://jhalderm.com/pub/papers/ivoting-ccs14.pdf. Accessed 13 Jan. 2019.

Goldwasser, S., et al. "The Knowledge Complexity of Interactive Proof-Systems." *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing - STOC '85*, 1985, doi:10.1145/22145.22178.

Goldwasser, Shafi, et al. *The Knowledge Complexity of Interactive Proof Systems*. Feb. 1989, http://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Proof%20Systems/The_Knowledge_Complexity_Of_Interactive_Proof_Systems.pdf.

Heiberg, Sven, et al. "Improving the Verifiability of the Estonian Internet Voting Scheme." *Lecture Notes in Computer Science*, 2017, pp. 92–107.

Introduction – Homomorphic Encryption Standardization. http://homomorphicencryption.org/introduction/. Accessed 14 Jan. 2019.

Lai, Wei-Jr, et al. "DATE: A Decentralized, Anonymous, and Transparent E-Voting System." *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2018, doi:10.1109/hoticn.2018.8605994.

Racanelli, Vito J. "Proxy Voting Is Broken and Needs to Change." *Barrons Online*, Barrons, 7

July 2018, https://www.barrons.com/articles/proxy-voting-is-broken-and-needs-to-change-15309 24318.

Rivest, Ronald L. *The ThreeBallot Voting System*. 1 Oct. 2006, https://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf.

Tsang, Patrick P., and Victor K. Wei. "Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation." *Lecture Notes in Computer Science*, 2005, pp. 48–60.

What Are Zk-SNARKs? https://z.cash/technology/zksnarks/. Accessed 14 Jan. 2019.

Wu, Wei-Jr Lai Ja-Ling. *An Efficient and Effective Decentralized Anonymous Voting System*. 18 Apr. 2018, http://arxiv.org/abs/1804.06674.

Yu, Bin, et al. "Platform-Independent Secure Blockchain-Based Voting System." *Lecture Notes in Computer Science*, 2018, pp. 369–86.

Zcash Protocol Specification. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf. Accessed 13 Jan. 2019.

Annex A - Paillier Encryption System

Let's consider c the Paillier encryption of a vote v using a random value r and a public key (g, n):

$$c = (g^v * r^n) \mod n^2$$

Considering $Z^*_{n^2}$, g being a random element in c , it encrypts the vote v by raising basis g to the power of v and randomises to a random factor.
Given the public key and the encryptions of v_1 and v_2, the system can compute $v_1 + v_2$ without knowing v_1 or v_2.

Key Generation: $(sk_Paillier, pk_Paillier) := Gen_Paillier(K_len)$ will generate the secret key $sk_Paillier$ (which must be stored safely) and the public key $pk_Paillier$ (which will be published during the setup phase), with given length K_len in bits.

Encryption: $ENC_Paillier(\zeta, pk_Paillier) \rightarrow C$ Let $\zeta \in Z_n$ be the plaintext to be encrypted. Select random $r \in Z^*_n$ and compute $C = g^\zeta * r^n \mod n^2$

Decryption: $\zeta := DEC_Paillier(C, sk_Paillier)$ Let $C \in Z^*_n$ be the ciphertext, compute plaintext as $\zeta = (L(C^\lambda \mod n^2) * \mu) \mod n$

ZK Proof-of-Result proves that a message is a member of a set of 1 out of n messages. In (Baudron et al.) the author proposes an efficient method to prove that, where: $\{v_j, e_j, u_j\}_{j \in P} := ZKPoR_{mem}(C, \Upsilon)$.

1) P picks k randomly from $Z^*_n, \rho - 1$ values $\{e_j\}_{j \neq i}$ in Z_n and $\rho - 1$ values $\{v_j\}_{j \neq i}$ in Z^*_n. Then computes $u_i = k^n \mod n^2$ and $\{u_j = v_j^n (g^{\zeta * j}/C)^{e_j} \mod n^2\}_{j \neq i}$

2) Prover computes $e_i = \sum_{k \neq i} e_k \mod n$ and calculates $v_i = \rho * r^{e_i} * g^{(e - \sum_{k \neq i} e_k) \div n} \mod n$ and sends the result $\{v_j, e_j, u_j\}_{j \in P} := ZKPoR_{mem}(C, \Upsilon)$ to the verifier.

3) Verifier checks $e = \sum_{k} = 1^{P} * e_k$ and $v_k^n = u_k(C/g^{\zeta * k})_k^e \mod n^2$ for each $k \in P$

Decryption of ZKPoR (Yu et al.): We define $(\delta, r) := ZKPoR(C, sk_Paillier)$, which will compute the plaintext δ and the random r to the ciphertext C, with the given secret key $sk_Paillier$. Compute $r \in Z_n^*$, denote $g(r) = r^n \mod n^2$.

Annex B - Ballot format and tallying

This voting system can be used in the majority or preferential voting systems. To achieve that, instead of casting votes on a *boolean* format, the cast is on an *array* of *integers:*

Example for a race:

Considering n candidates in the array $[Candidate_0, Candidate_1, Candidate_2, ..., Candidate_n]$ of a specific race.

The ballot B will always be initialized with 0 in all indexes.

$$B = [0, 0, 0, ..., 0_n]$$

For casting a vote on a majority system

Consider the chosen candidate receives 1 and all the rest receives 0:

If the $Candidate_1$ received the vote, the unencrypted ballot B will be filled with the values as follow:

$$B = [0, 1, 0, ..., 0_n]$$

For casting a vote on a preferential system

The most preferred candidate will receive a max note m, continuously decreasing 1 in $m = (m - 1) \in m > 0$ to the next most preferred candidate, until $m = 1$. All other candidates will be filled with 0:

If you need to choose max 3 candidates, being the $candidate1$ as the most preferable candidate, it will receive the max value 3, followed by the $candidate0$, who receives the

$m-1$ value 2 and then by the $candidate3$, who receives the $m-2$ value 1. All the not chosen candidates will be filled with value 0.

The unencrypted ballot B array will be filled with the values as follow:

$$B = [2, 3, 0, 1, ..., 0]$$

Each vote in the ballot B will be encrypted as described in Annex A.

Tallying the votes

The process for tallying the votes is the result array R with the sum $\sum$ of all the votes for each candidate.

$$R = [\sum_{Candidate_0} (voter_0 + ... + voter_n), ..., \sum_{Candidate_n} (voter_0 + ... + voter_n)]$$