Assembly Voting X

The election system was designed with a focus on security and verifiability. All cryptographic algorithms are inspired from academic papers carefully bonded together to form our protocol. Citations to academic articles are provided for an in-depth understanding of the algorithms.

The design of the system is modular, that makes it very easy to configure in order to reach the desired properties of your election. Also, in case an updated algorithm is developed, it is very easy to replace a particular module with an updated version.

The current document is structured in the following way. First, we describe the functionality of each component that makes up the election system. Next, we present the election process including all different phases and different roles that are involved in the process. In the third chapter, we state what security properties the system achieves. We explain how these properties are achieved and what components are responsible for each property.

In conclusion, we present a table that compares Assembly Voting X to some existing solutions from the market. The table also contains a comparison to the current postal voting system.

Cryptographic components

Cryptosystem

We are using a fast and secure cryptosystem based on elliptic curve cryptography. Particularly, we are using a highly secure and documented elliptic curve, called secp256k1, also used in the Bitcoin transaction system. Our encryption mechanism is based on the very popular algorithm called ElGamal encryption scheme that entails the use of a private-public key infrastructure.

All voter choices are encrypted and sent over the network to the election server with no possibility of eavesdropping. We will refer to an encryption of a choice as a ballot.

With this configuration, the vote that needs to be encrypted can be approximately 30 bytes in size. This would be enough for a referendum, a simple election, a multiple-choice election with maximum 30 choices, an STV election with maximum 15 choices or a write-in vote with maximum 30 characters. If more data is required, the ballot can be configured to encode multiple cryptograms. We will try, as much as possible, to configure every election to fit in a simple ballot of one cryptogram.

Threshold decryption

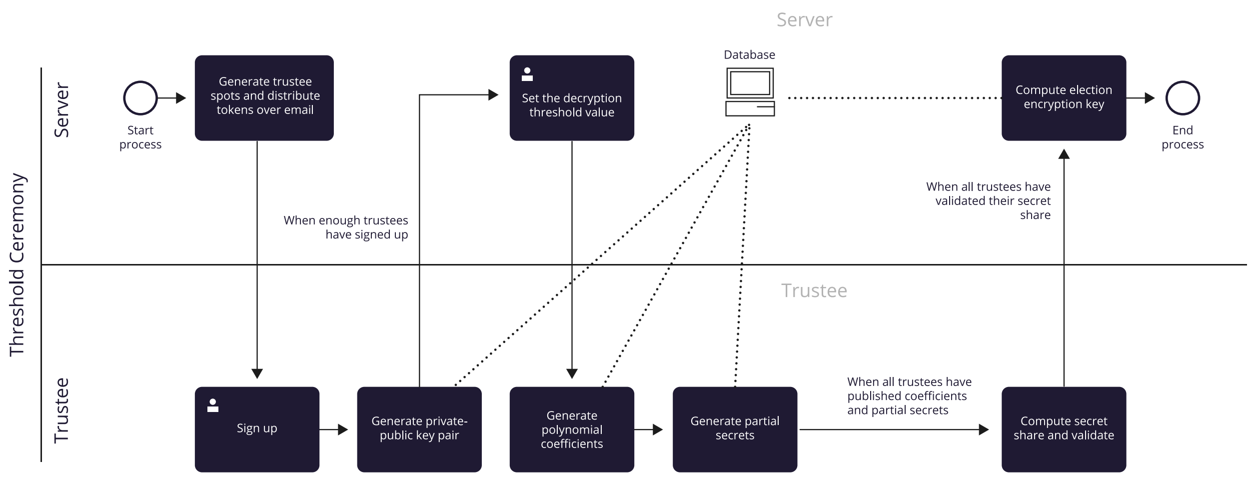
To defend against a single point of failure, the private key of the election is split into several parts, each in possession of different people of the election committee, which we will refer to as trustees. In order to decrypt the result of the election, a certain threshold of trustees have to participate, otherwise decryption is not possible. This brings us two benefits:

- In case of a trustee loses her share of the private key, the results could still be decrypted as long as the threshold of trustees can be met.
- In case of a corrupt trustee, results cannot be manipulated as long as a threshold of trustees are honest.

The election system uses an "e-out-of-n" threshold decryption system presented in the academic paper "A Threshold Cryptosystem without a Trusted Party" written by professor Torben Pryds Pedersen at the Aarhus University in Denmark [1]. The paper is based on other academic articles that explain the mathematical principles of the threshold cryptosystem [2] [3]. The system needs at least e trustees to collaborate out of all n in order to decrypt the results (e.g. 3 out of 5 trustees). Parameters are fully configurable, but it is recommended that the threshold is at least half of the total number of trustees.

Before election starts, all trustees have to participate in a threshold ceremony where they exchange cryptographic data used for generating the election public key. During this process, each of them computes their own share of the private key, that must be used to decrypt the result of the election. All actions taken by trustees come with proofs and can be publicly verified that are correctly computed.

Note that during this ceremony, nobody is able to compute the entire main private key associated with the election public key. This means that nobody has the power to decrypt results alone. All mathematical procedures that trustees have to follow are described in the academic paper.
An overview of the threshold ceremony can be seen in the picture below.

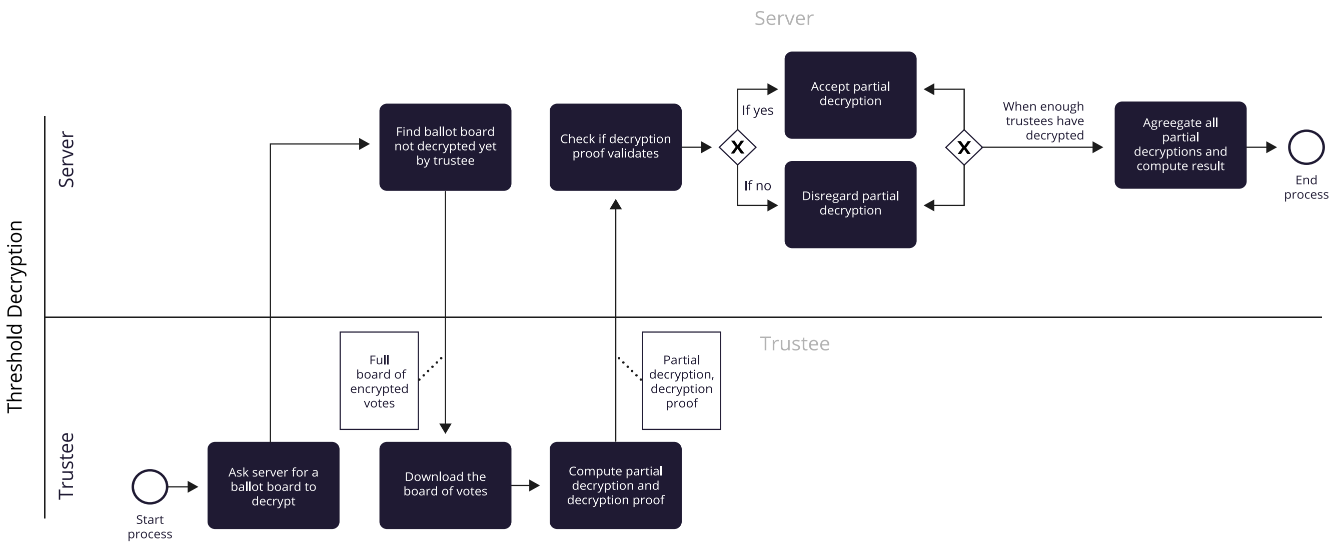


All trustees have to securely store their share of the private key until results can be decrypted.

During the decryption phase, trustees have to compute a partial decryption of the entire ballot board using their share of the private key and generate a proof of correct computation. Each trustee publishes her partial decryption and proof to the election server, which will accept it if the proof validates. Note that the validation is publicly accessible.

The proof of a partial decryption consists of a list of Discrete Logarithm Equality Zero-Knowledge Proofs, one for each cryptogram from the ballot board. An optimization of this has been implemented as described in the paper "Zero-Knowledge Argument for Simultaneous Discrete Logarithms" published by professor Shermann Chow et al. at the Courant Institute of Mathematical Science New York University in USA [4].

When enough partial decryptions have been received (threshold limit was reached), the election server can aggregate all partial decryptions in order to extract the results of the election. Again, the mathematical procedures are explained in the academic paper [1].
The overview of the threshold decryption can be seen in the diagram below.



Voter Credentials distribution

Voters receive their credentials via one or multiple channels from different Credential Authorities that work independently from our system. Each Credential Authority should use a distinct communication channel for distributing credentials (sending letters, e-mail, SMS).

Voter credentials are generated randomly as a private-public key. The voter receives the private key which will be used as a signing key, while our server receives the associated public key, which will be used as a signature verification key. It is very important that our server does never come into possession of voters' signing keys because it must not be able to replicate a voter's digital signature.

When authenticating to the election system, a voter has to input all credentials received from all Credential Authorities.

In case there is only one Credential Authority, it is obvious that it knows all credentials of all voters and it might, potentially, launch a large-scale attack impersonating every voter. To avoid such a single point of failure scenario, we recommend having multiple Credential Authorities to generate voter credentials, using distinct communication channels for distributing them. In this case, a large-scale attack is infeasible as long as there is at least one honest Credential Authority.

Digital Signature

To preserve the integrity of a vote, each cryptogram is accompanied by a digital signature that certifies that the value of the cryptogram is genuine and can never be modified. Moreover, a digital signature certifies the correlation between a voter and her ballot.

A digital signature is generated using the Schnorr Signature Algorithm described in the academic paper "Efficient identification and signatures for smart cards" written by professor Claus-Peter Schnorr [5]. Voter's credentials are used as signing key in the signing algorithm.

Once the cryptogram is published next to its signature, it is impossible to change the value of the cryptogram because doing so will invalidate the signature, thus mitigating the possibility of a misbehaving server.

Vote Confirmation

After the voter submits her vote (in form of a cryptogram), the server will send back a confirmation (receipt) that her vote has been received in form of a signature on the vote information. One might say it is similar to the Digital Signature protocol, but this time it is the server who signs and confirms the arrival of the vote.

The voter will have the option of saving the receipt on a personal computer. Based on it, the voter will be able to check that her vote is included in the public bulletin board.

Please note that, this receipt proves only the fact that the voter has voted. It does not prove the way she voted. Thus, the vote confirmation protocol does not violate the receipt-free property of the election that says that the voter should not be able to prove to a third party the way she voted.

Public Bulletin Board

During the voting phase, all ballots are published on an append-only list, called the public bulletin board. All voters have access to this list in order to verify that their ballot has been registered as cast.

When a new ballot arrives on the bulletin board, a new hash value is associated to the new state of the board. The value is computed by applying a hash function on the information of the new ballot appended to the hash value of the previous state (before the new ballot was registered).

Each voter has the possibility of validating whether her vote is included on the board or not, using her vote confirmation received from the server. The system will point the voter to her particular vote from the board and she can validate that no data has been tampered with. Note that during this process, the voter validates both that her vote is included and that the integrity of the entire board has been maintained.

In case the hash value of the vote confirmation does not match the hash value of the vote from the bulletin board, it represents an attack to the integrity of the bulletin board (a vote has been removed or replaced). Thus, an inside attack to the integrity of the board can be easily intercepted.

Encryption Protocol

Instead of the voter encrypting her vote by herself, we propose a scheme where the voter and the election server collaborate in order to generate a cryptogram. The process starts by the server delivering an empty cryptogram to the voter. The latter will encrypt her vote on top of the empty cryptogram received. In this context, the randomness used in the generation of the final cryptogram, is shared between the voter and the election server with no single party knowing the entire value.

The empty cryptogram sent by the server has to be accompanied by an Interactive Zero-Knowledge Proof of Discrete Logarithm Equality to confirm that the cryptogram is indeed empty. The reason it needs to be interactive is that the proof (that the initial cryptogram was empty) needs to convince the voter only, therefore it needs to not be universally valid. Together with the encryption of her vote, the voter also sends a Zero-Knowledge proof that the empty cryptogram was used in the encryption process.

If the voter tries to convince a third party about the way she voted, she can prove her vote based on the initial cryptogram received, but she cannot prove that cryptogram empty. Hence, the protocol is receipt-free.

By default, the voting application will hide the randomness used in the encryption so a regular voter cannot prove the way she voted. Nevertheless, a malicious voter with enough hacking skills could trick the voting application into revealing this sensitive information.

Though, by following our encryption protocol, a malicious voter could still not prove the way he voted because part of the encryption was generated on the election server. Our system is receipt-free as long as the attacker is not in control of both the voting application and the election server.

Mixnet

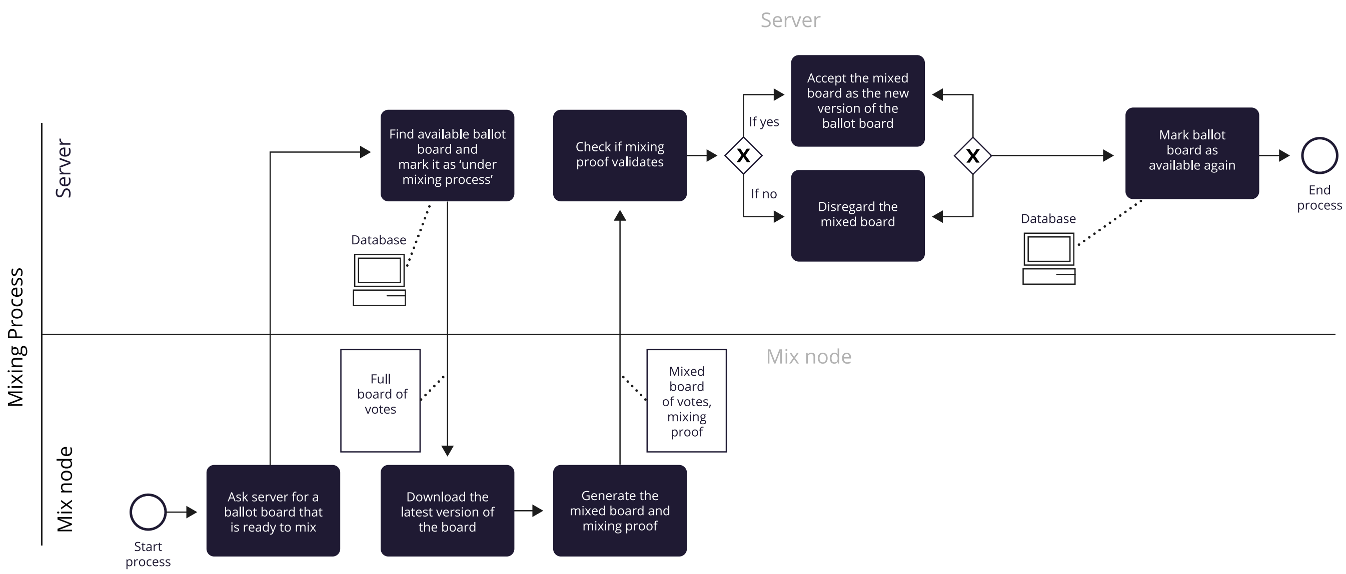
To preserve anonymity, the link between a voter identity and his ballot has to be broken. In our election system, we achieve that by passing the entire ballot board though a mixnet, formed of several mix nodes. Each mix node applies a re-encryption algorithm on each cryptogram from the board and shuffles them in a new random order to form the new version of the ballot board. In addition, a proof of Correct Shuffle is generated to validate the correct re-encryptions of the original ballots.

The proof has been inspired from the academic paper called “An Efficient Scheme for Proving a Shuffle” published by researcher Jun Furukawa from Internet Systems Research Laboratories at NEC Corporation, Kawasaki Japan [6]. The paper has been reviewed and updated many times over the years in different articles [7] [8]. All cryptographic procedures involved in the generation and verification of the proof are described in the paper.

Mix nodes apply their mixing procedure in sequential order, meaning that each mix node mixes the ballot board that the previous mix node has outputted. The first mix node mixes the initial, original ballot board. The final version of the ballot board is the one that the last mix node computes.
In case one proof of shuffle is invalid, that mix node is removed and the process resumes from the previous valid result.

All mix nodes are responsible for safely storing their mixing parameters used in the generation of the board. In case of a corrupt mix node that publishes his mixing parameters, our system still preserves anonymity as long as there exists at least one honest mix node.

An overview of the mixing process can be seen in the picture below.



ASSEMBLY VOTING

Spoiling Ballot feature

After encrypting her vote (generating her ballot), the voter has the choice either to commit to her ballot and register it on the ballot board or to challenge the encryption mechanism and verify what the ballot actually encrypts (spoil the ballot).

If spoiling ballot, it will be printed on the screen both the ballot and the randomness used to encrypt the ballot (QR code format). The voter uses a second device to scan these values and to decrypt the content of the ballot. If the content of the ballot does not correspond with her choice, her voting device might be compromised, as an attacker might trick the voting application to encrypt different values. Otherwise, the voter gains confidence that the voting device outputs genuine ballots.

The second device, used for verification, can be a mobile phone with an app installed that is able to perform basic cryptographic operations. This device might be completely off-line for the voter to gain confidence that it is not manipulated by an attacker.

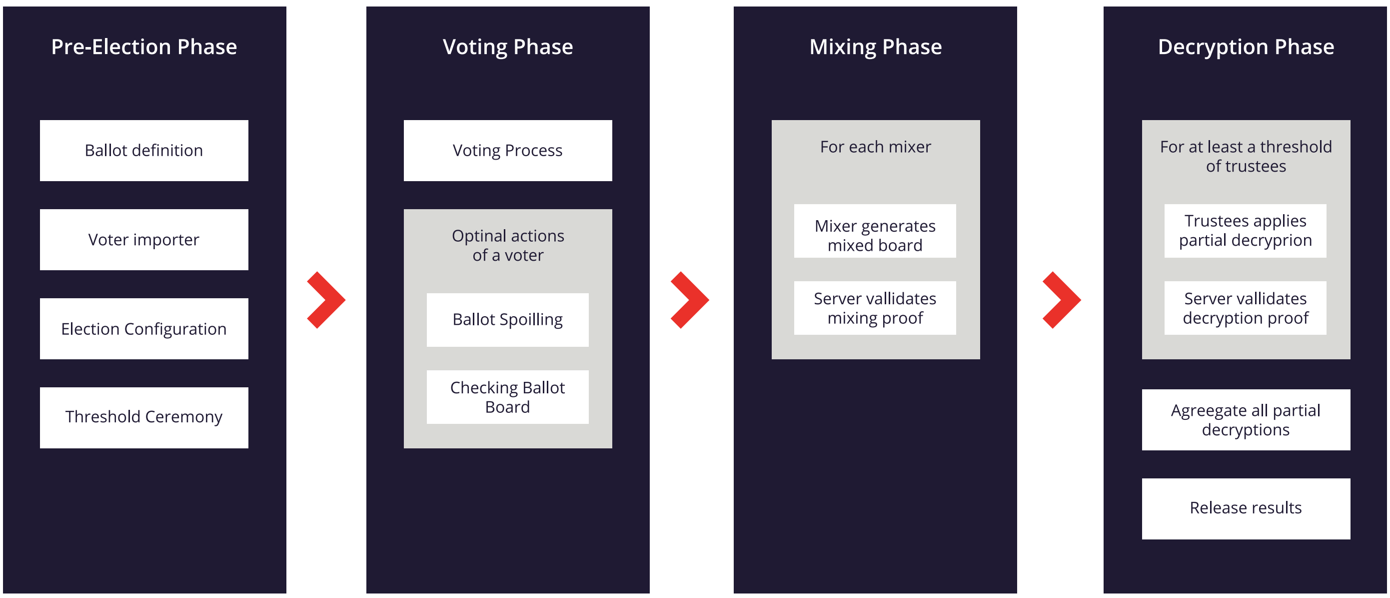
Because it has been decrypted, the spoiled ballot cannot be used anymore so the voter has to re-vote. This process can be repeated as many times as needed, until the voter gains enough confidence in her voting device.

If committing to ballot, the election system will register it on the ballot board and the voting application will erase the random number used in the encryption. The voting process is finished.

One might say that a malware can be programmed to interfere with the voting application only on its second or third try, but there is no certainty on how many times each voter may try to spoil her ballot. This way, we argue that an attack to the voting devise will get caught with high probability.

Election Process

The overview of the entire election process is available in the diagram below. Descriptions for each step follow afterwards.



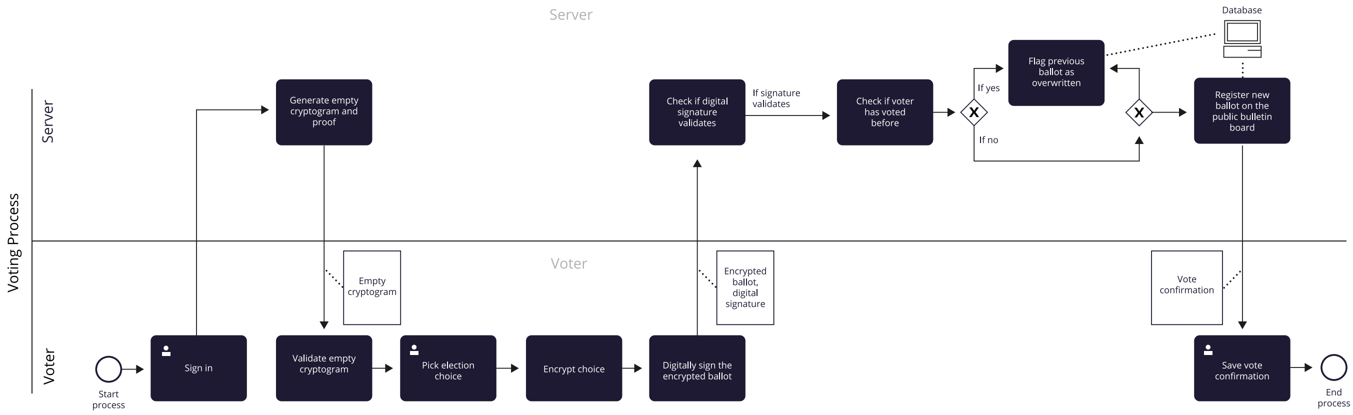
Pre-election phase:

- The election system has to be provided with a list of eligible voters. Each voter must have valid contact information for each communication channel of the Credential Authorities. The election administrator is fully responsible for providing an accurate voter list and valid contact addresses.
- The Credential Authorities generate voter credentials and distribute them over particular communication channels. They also submit voters signature verification keys to the election system.
- The election trustees have to participate in the threshold ceremony in order to generate the election encryption key. Each trustee is responsible for securely storing their share of the election decryption key.

Voting phase:

- The voter has to login to the system, using credentials received by email.
- The voter selects her choice of candidate and confirms it.
- The voter is presented with her encrypted ballot in a readable form (Hex / Base64 string or QR code). This should be written down (or saved) for further verification.
- If spoiling ballot feature enabled:
    - The voter has the option to verify that the encrypted ballot contains the actual selected choice.
    - The voter will be presented with her ballot in encrypted form (as a QR code) and the random number used in the encryption process (a second QR code). The voter has to scan both QR codes with a second device (a mobile phone), that will perform the decryption of the ballot, revealing voter's choice. The second device can run offline so it cannot be interfered by an attacker.
    - This process will invalidate the ballot, as it was decrypted, and the voter will be asked to vote again.
    - The voter can repeat this process as many times as needed until she gains confidence that her choice is encrypted correctly (the vote is cast as intended).
    - In case the ballot decrypts to a different value than expected, this shows a sign of attack to the client application.
- The voter generates a digital signature on her ballot.
- The voter submits her encrypted ballot and the signature to the central server.
- The voter receives and saves the confirmation that her vote has been registered.
- The voter can check the public bulletin board that it contains her encrypted ballot (by typing the value of the encrypted ballot or by uploading the confirmation receipt). This way, the voter gains confidence that her vote is registered as cast.
- The voter is able to register more ballots, during the voting process, out of which only the last one will count. The previous ballots become overwritten.

The overview picture of the voting process is available below.



After voting:

- All the invalid and overwritten ballots are removed, and the bulletin board is sealed. This contains all votes that should be counted.
- Mixing phase

    - **The bulletin board passes through the mixing phase that will shuffle the order of the ballots in an indistinguishable way. The entire mixing phase is split amongst multiple mix nodes that apply their shuffle sequentially. Each mix node provides a mathematical proof that certifies that no content of that ballots has been tampered with.**
    - **Any observer is able to verify these proofs and gain confidence that no content of the bulletin board was altered in the mixing process.**
    - **After mixing phase, the piece of information regarding the connection between an identity and its ballot is shared amongst all mix nodes. They are responsible for securely storing their shuffle configuration.**

- Decryption phase

    - **The ballot board outputted by the last mix node is the ballot board version to be decrypted.**
    - **A threshold of trustees has to participate in the decryption phase. Each of them is computing a partial decryption of the bulletin board together with a mathematical proof of correct computation.**
    - **All partial decryptions together with their proofs are made public so any observer is able to verify the correctness of the process.**
    - **When enough partial decryptions have been submitted, the content of the ballots can be extracted from the bulletin board by aggregating all partial decryptions. This aggregation process is publicly computable, thus accessible to an observer.**

Results:

- After the raw result has been published (list of all votes), the final result has to be computed, according to the election type (referendum, simple election or STV), and the winner has to be announced.

Proporties

Individual Verifiability

The voter can see and save the encrypted ballot generated on her computer. If the ballot is registered, the voter is given a receipt that confirms that her vote has been received. She can, further on, check that it was correctly registered on the server by verifying that her encrypted ballot exists on the bulletin board.

If spoiling ballot feature enabled, the voter can check that her client application behaves correctly. After the voter selected her choice and the encrypted ballot has been generated, the voter is given the option to register the ballot or to spoil it.

If spoilt, the client application will show on the screen both the ballot and the random number used in its encryption. The voter can use a second device to decrypt the content of the ballot and verify that it corresponds to her choice. Having been decrypted, the ballot cannot be used anymore, so the voter has to cast another vote.

Each voter is recommended to use this feature, at least once, as a verification mechanism of their own system (computer).

There is no universal verifiable mechanism to check that all encrypted ballots published on the bulletin board have not been spoiled before. The server has to be trustworthy of correctly handling of encrypted ballots.

Universal Verifiability

During the voting phase, observers constantly monitor the content of the public bulletin board. At the end of the voting phase, all observers have to confirm the integrity of the board before it can move further to mixing phase.

After the ballot board has been cleaned and sealed (end of voting phase), all cryptographic operations applied on the set of ballots are publicly verifiable. Both mixing proofs and decryption proofs are published, and observers are allowed to verify.

While the individual verifiability is optional, the universal verifiability is mandatory. All mixing and decryption proofs have to be validated by the server to be included in the process.

During the mixing phase, validation of a proof is needed after each mix node before the process can continue with the next mix node. On the other hand, in the decryption phase, all partial decryption proofs can be checked at the same time, so all trustees can perform the decryption process simultaneously.

Eligibility Verifiability

Each ballot that arrives at the server is accompanied by a digital signature generated by its voter. All ballots are published on the public bulletin board together with their signatures. Any observer will be able to validate any digital signature associated to an eligible voter identity.

Moreover, each valid digital signature certifies the integrity of the vote because any tampering with a vote on the bulletin board will result in invalidating its digital signature.

Vote Secrecy

The secrecy of the ballots is enforced by ElGamal encryption. The threshold decryption scheme prevents anybody from reading a partial result before the decryption phase. Note that even the election server is not able to compute any results ahead of time.

On the other hand, the voting device learns the voter's choice. It is voter's responsibility to have a clean and secure environment with respect to malware, key loggers etc.

Anonymity

Anonymity is provided by breaking the connection between a voter and her vote. This connection, as a piece of information, is split during the mixing phase into several pieces, one for each mix node. If all mix nodes put their pieces together, the connection between all voters and their votes can be reconstructed, but as long as at least one mix node keeps his piece of information secret, the anonymity of the ballot board is preserved.

Analytics and Auditing

All kinds of analytics can be performed as the ballot board is publicly available.

On the other hand, auditing particular ballots works exactly against the anonymity property of our election system. In principle, auditing can be performed but it requires cooperation of all mix nodes. This process should be allowed only to certified scrutineers.

Tamper Detection

Tamper detection happens on two levels:

- Server side: Because of voters constantly checking their vote confirmations, tampering (deleting or modifying) with the ballot board is immediately detectable.
- Voter side: Tampering with the voting application is detectable through ballot spoiling process.

Coercion resistance

The election system provides coercion resistance to a certain extent. If the receipt-free feature is enabled, a voter is not able to provide evidence on the way she voted to a third party e.g. a coercer. Vote copying is mitigated as well because the voter is not performing the encryption of her choice by herself (election server is involved in the encryption process).

Our system is coercion resistant as long as:

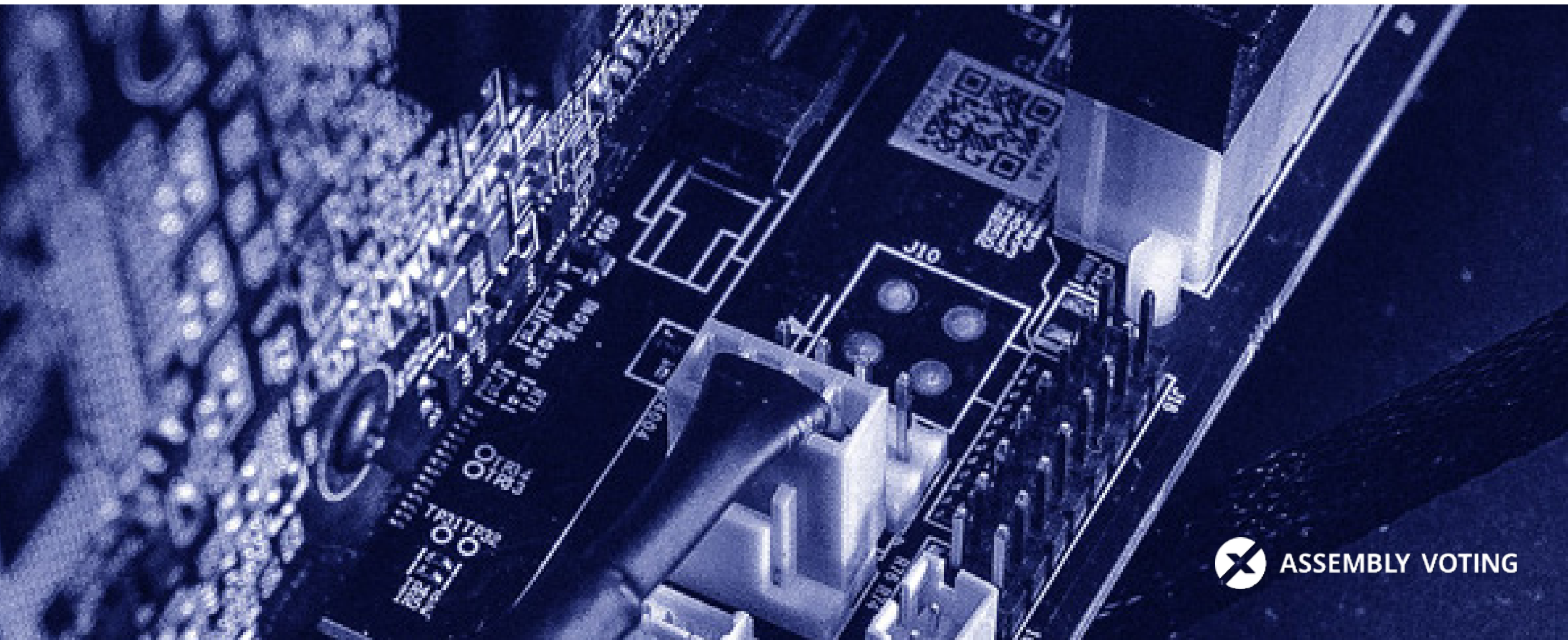
- The coercer does not sit next to the voter and see the voting process
- The coercer does not control the election server

Receipt freeness

Following our encryption protocol, the voter cannot prove to a third party what the content of her ballot is. Because the election server participates in the encryption process (by submitting an empty cryptogram), the voter has to output the following proofs for convincing a third party about her vote:

- Proof of her encryption
- Proof that initial cryptogram received from server is empty

The first one is trivial. The second one is impossible because the voter is able to fabricate a different valid proof based on any values. This means that a voter is able to lie about her vote by generating a valid proof for claiming that. This makes her proof



Property	Postal voting (current system)	Building society	Hellos	Estonian system	CH Vote	Assembly Voting X
Provides means to collect and maintain accurate registered voterslist, and means to ensure equivalence with actual voters	No, such a list must be provided as in-put to the system. and scrutineer must ensure actual voters correspond to the list.	No, ditto.	No, ditto.	No, ditto.	No, ditto.	No, ditto.
Provides secure means to authenticate voters	Yes, assuming that the postal system can be relied on.	Yes, code numbers received through the post (or email).	Yes, user name and password received through email.	Yes, by government-issued smart card.	Yes, code numbers on Voting Cards received through the post.	Yes, election codes received by email or post.
Provides detectability of attacks to server	Not applicable. There is no online server.	No.	Has strong verifiability properties..	Has weak verifiability properties.	Has strong verifiability properties.	Yes, - by constantly checking the consistency of the public board - by the voter checking that her vote was registered as cast.
Provides detectability of attacks to client (voting application)	Not applicable. There is no voting client.	No.	Yes, by involving two different client platforms.	Yes, by involving two different client platforms.	Yes, by exchanging codes as part of the vote casting process.	Yes, through spoiling ballot feature (involves a second client platform).
Allows analytics	Yes.	Yes.	No. Could possibly be extended to allow.	Unknown. Probably not, but could possibly be extended to allow.	Unknown. Probably not, but could possibly be extended to allow.	Yes, needs mix nodes cooperation.
Provides verifiability (reducing dependence on scrutineer)	No.	No.	IV + UV (but not EV).	IV only.	IV + UV (but not EV).	- IV (verifying registered as cast and possibly, cast as intended). - UV (verifying mixing proofs and decryption proofs). - EV (by digital signature infrastructure).
Provides ballot secrecy from third party	Yes, but weak (relies on secrecy of post).	Yes.	Yes.	Yes.	Yes.	Yes, by Elgamal encryption.
Provides incoercibility from third party attacker	No.	No.	No.	Yes, by re-voting.	No.	Yes, to some extent, In terms of receipt-free. No, as long as the coercer sits next to the voter and sees the voting process.
Provides incoercibility from attacker that controls election system	No.	No.	No.	No.	No.	No.

ASSEMBLY VOTING

Bibliography

[1] T. P. Pedersen, A Threshold Cryptosystem without a Trusted Party, Aarhus: EUROCRYPT '91.

[2] Y. Desmedt and Y. Frankel, Threshold cryptosystem, Milwaukee: EE & CS Department University of Wiscounsin-Milwaukee, 1990.

[3] A. Shamir, How to Share a Secret, Massachusetts Institute of Technology, 1979.

[4] S. Chow, C. Ma and J. Weng, Zero-Knowledge Argument for Simultaneous Discrete Logarithms, New York: COCOON 2010.

[5] C.-P. Schnorr, Efficient identification and signatures for smart cards, New York: CRYPTO 89, 1989.

[6] J. Furukawa and K. Sako, An Efficient Scheme for Proving a Shuffle, Kawasaki: Crypto 2001.

[7] J. Furukawa, H. Miyauchi, K. Mori, S. Obana and K. Sako, An Implementation oa a Universally Verifiable Electronic Voting Scheme Based on Shuffling, Kawasaki: Internet Systems Research Laboratories, NEC Corporation, 2002.

[8] J. Furukawa, Efficient, Verifiable Shuffle Decryption and Its Requirements of Unlinkability, Kawasaki: NEC Corporation, 2004.