In this chapter, we present our own solution to the electronic election system and the final voting scheme we implement. Then, we bring some description of existing security schemes and services, we use in our solution. We also discuss some other proposals of the voting scheme, which we modified or rejected.

1 Players in Our Voting Scheme

We shortly describe each player in our voting scheme and their purpose.

- **Voter** V authenticates to S using *authentication authority* T and casts their vote using *voting application* A. They also can cast their vote in paper form in a special-purpose room.
- **Electoral board** B are in charge of key generation during the first phase. They hold secret key SK generated for *machine for counting votes* M and share corresponding public key PK. They also securely transfer encrypted data from *server for vote collection* S to M. Data can be then decrypted using SK, provided to M by B. In the end, they publish the result of the elections. Last but not least, they are responsible for paper votes casted in a special-purpose room designed for the elections.
- **Database** D stores information about all the candidates and identities of all authorised voters. This information is then used by A and S to display list of possible candidates and authorise *voter* V. This database is physically stored on server S and is accessible only to S.
- **Voting application** A provides user interface for V in which they can cast their vote. It also validates the vote, converts it to a proper format and encrypts it using PK. Then it sends the vote together with the voter's identity.
- **Server for vote collection** S receives encrypted vote and voter's identity and checks whether this identity corresponds to any of those stored in *database* D. If the encrypted vote comes from an authorised voter, then it stores the encrypted vote and the voter's identity in its internal database of votes. This server also provides storage for voting application A and database D.
- **Machine for counting votes** M receives all the votes (without identity of any voter) stored in S's internal database and secret key SK. It, then, decrypts all the received votes using SK, validates them and computes the result of the elections, which is finally published. We point out that this machine is not connected to any network prior or during the elections. This machine is restricted to be in off-line mode until the results of the elections are published and it is proved that the election process has been correctly performed.
- **Authentication authority** T is responsible for authentication of V and provides M with V's identity, which must *1-to-1* correspond to a value in D. For purposes of our voting scheme, we use *Cosign*, which is a service commonly used for authentication of students studying at our faculty. From this point, we mean *Cosign* any time we talk about T. Similarly, we

use term *UK login* of voter V instead of *V's identity*. More information about *Cosign* is provided in section **??** and chapter **??**.

2 Our Voting Scheme

Bearing in mind that we require the voting process to be run in an Internet browser and that this technology has several limitations, we propose the voting scheme described in this section. The reader, then, can read more about limitations of the technology used in chapter **??**.

Our voting scheme is designed to have three subsequent phases: *initialisation phase*, *voting phase* and *counting phase*. The basic purpose of these stages is discussed in chapter **??**. In this section, we provide a detailed description of what is done during each of these phases.

2.1 Initialisation

1. Database D with a table of candidates is created and stored on S. For each candidate, the table contains a unique ID and name of the candidate.
2. a table of authorised voters is created in D. For each voter, this table contains their *UK login* and a boolean variable set to 0. This variable indicates whether the voter has already casted a paper vote.
3. a special *UK login* is created for the electoral board B. This login is used when a voter casts their paper vote.
4. Public key PK and secret key SK are generated. They are to be used to encrypt the vote by the voting application A and decrypt it by the machine for counting votes M, respectively.
5. Using *Shamir's Secret-Sharing Scheme*, the key SK is divided into n parts, where n is the number of members of B. The key SK can be reconstructed from any of combinations of k parts, where k is the smallest number of members of B that have to sign the *Protocol of elections*. The reader can read more about the scheme in section **??**.
6. Public key PK is shared with all the voters by inserting it into A as a constant value for the whole elections.

2.2 Voting

This phase follows after initialisation was completed.

for each voter V:

1. Voter V opens a specific Internet location in their browser and logs in using *Cosign* in order to authenticate to S.
2. If V is successfully authenticated, voting application is launched in their Internet browser. This application contains a form in which there are three options for each candidate.

3. When V submits the form, it is then validated regarding the rules of the elections. When form valid, vote in the specific format is created and this vote is encrypted using *S/MIME* protocol and PK. Finally, the vote and voter's *UK login* are sent to S via an *HTTPS* connection.

4. When S receives the encrypted vote, it compares the sender's *UK login* with logins in the table of authorised voters stored in D. If it matches exactly one record, then the login and the encrypted vote are stored in S's internal database of votes. If there already is a record with the same login and vote different form the special `0` character vote in the internal database, the former vote is rewritten with the new one.

5. If V decides to use paper vote and goes to the special-purpose voting room, the electoral board B authenticate to S using their special *UK login*. When they give V a ballot paper, they access the table of authorised voters and set the boolean value in the voter's row to 1. When V puts their vote to the ballot box in the room, B sends a special `0` character vote together with V's *UK login*, which rewrites V's electronic vote if they have sent one.

2.3 Counting

1. Server S is disconnected from network and off-line mode is enabled.

2. Using a secured USB device, the electoral board B transfers encrypted votes from S's internal database of votes to machine for counting votes M. It is crucial that *UK logins* stored in this database be excluded from the transfer.

3. With aid of at least k members of electoral board B, secret key SK is reconstructed. Then, it is inserted into M and used to decrypt votes.

4. Decrypted votes are, then, validated by M. All votes in the right format that follow the rules of the elections are counted and the final result is published.

3 Format of the Vote

Definition of format to which each vote is formatted prior to being sent to the server for vote collection S is also part of our voting scheme. We want the vote to be stored in one compact file and the choices to be clearly separated. It is known how many candidates are in the elections. Let M be the number of candidates. Hence, we propose that the vote is of the following format:

```
<candidate_number_1#value_1>...<candidate_number_M#value_M>
```

Here, `M` is the representation of number of candidates M.

This format represents a string of tuples (c_i, v_i), where $c_i \in \{1, ..., M\}$ is `candidate_number_i` and $v_i \in O$ is `value_i`, $i \in 1, ..., M$. Value v_i is numerical representation of one of the voting options and O is the set of all possible numerical representations. For example, 1 represents YES, 0 represents NO and $O = \{0, 1\}$.

Let $(c_1, ..., c_M)$ be an ordered set of all candidate numbers. Then, $(c_1, ..., c_M)$ is a permutation of $\{1, ..., M\}$.

It is obvious that a candidate number 1, for instance, is given YES if and only if the vote contains exactly one tuple of value $(1, 1)$.

4 Cryptography Used in Our Solution

In our voting scheme, we used these cryptographic protocols and schemes to securely transmit and store the data.

4.1 S/MIME

4.2 Shamir's Secret-Sharing Scheme

4.3 Cosign

5 Accomplishment

In this section, we discuss how our solution meets requirements.

6 Other Proposed Solutions

In the course of designing the final voting scheme for our election system, we proposed some other solutions. Those were either rejected, or the final scheme was based on them and their modifications. There are two elementary design classes being in consideration during the whole process. Those are based on question whether server gives client any additional information used to authorise the sender of the vote. Regarding this, two meaningful classes are considered: *authorisation with a token* and *authorisation with user information.* In all of the schemes we considered, at least three players represented by computers are needed: a *client* C, an *authentication server* A and a *counting machine* B.

6.1 Authorisation with a token

In this design, anonymity is achieved by using a *token.* This token assures B that the vote comes from an authorised voter without associating it with a particular voter. In chapter **??**, we describe some of existing solutions using this design.

Here, we briefly describe a scheme that uses server A's signature to prove that the vote come from an authorised voter. Let V be voter.

1. Voter V logs in and voting client application is opened.
2. Client C sends $n \in \mathbb{N}$ encrypted votes to A, where 1 of those is the real vote and the other votes are fake.
3. If V is an authorised voter, A signs all n votes and sends them back to C.
4. Client C sends the signed real vote to B.
5. Server B validates the votes and if the signature