INRIA Nancy Grand est

Improving the time complexity of the Civitas Protocol

Author:
Akash Garg

Supervisor:
Dr. Steve Kremer

July 10, 2015

Acknowledgement

Simply put, I could not have done this work without the lots of help I received from the entire CASSIS Research Team at INRIA. The work culture at INRIA is really motivating. Everybody is such friendly and cheerful companion here that work stress never comes in way.

I would like to express my gratitude towards Dr. Steve Kremer for providing the nice ideas to work upon. Not only did he guided about my project but he always encouraged me to discuss ideas with him, corrected my mistakes and also helped me so as to make my adaptation smooth. Without those discussion it would have been impossible to complete the work in such a manner.

I would also like thank all the members of the CASSIS team for making my stay so pleasant in France.

Author

Preface

This report documents an attempt to improve the time complexity of the Civitas E-voting protocol done as a part of summer internship at CASSIS Research Team, Inria Nancy Grand Est , France under the supervision of Dr. STEVE KREMER. The report first gives all the back ground information about Civitas protocol and its theoretical implementation and time complexity of the implementation. Then we have explained the suggested changes in the implementation and the effects on the time complexity of the protocol. The main goal is to make the civitas more usable for elections with huge number of voters.

The various aspects of the protocols have been explained in an abstract sense so as to make it easier to understand. I have tried my best to present technical details in as simple manner as possible. I hope to succeed in my attempt. Any questions or suggestion about this work are most welcome.

Akash Garg
akash.garg@inria.fr

1 Introduction

Our world has changed a lot in the last two decades. Technology has impacted almost every sphere of life. With all these advancements everything has changed a lot or is undergoing massive changes. One of the most important aspect of a democratic world, elections are also undergoing massive changes. Currently we are moving from traditional voting methods to new Electronic voting. Some of the elections where e-voting has been used are Australian elections in 2007, general and municipal elections in Belgium, all Brazilian elections and some municipal elections in Canada. E-voting is making fundamental change in the foundations of democracy as the entire process of elections will be changed.

In the recent past, many efforts have been made to make the transition from traditional voting to e-voting. One of the major field of research is the e-voting protocol to be used. Developing a protocol for e-voting has many challenges like end-to-end verification, voter privacy, coercion resistance etc. Many of the protocol have been developed over the years to tackle these issues.

Civitas is on the forefront of those protocols. It provides all the above features. It provides end-to-end verification (it can be verified that each legitimate vote has been counted and the result is correct according to the votes in the ballot boxes) and coercion resistance (preventing adversary from convincing a voter to vote for a particular candidate) while maintaining voter privacy (the candidate for which a voter votes cannot be known). But still this protocol has only been used in university elections and not in general elections. One of the major limiting factors in the time complexity. Civitas has $O(m^2)$ time complexity not suitable for large number of voters. Here m is the number of votes cast.

In this report we have devised a method which makes significant improvement in the running time of Civitas thus making it more usable for large scale elections. We have achieved a time complexity of O(mlogm) for the working of civitas.

2 Civitas

Civitas is the first voting protocol that provides coercion resistance and end-to-end verifiability by ensuring voter and universal verifiability. It preserves

voter privacy and ensures that a voter cannot prove to the adversary whom he voted for. Thus Civitas is one one of the most suitable voting system for remote voting.

Civitas is the first voting system to implement a scheme proved to satisfy coercion resistance and thus taking the secure electronic voting to reality.

2.1 Security Model

Remote Voting: It allows the voter to vote from any location thus making it very convenient for him to vote. Remote voting is thus much more general and harder than any form of supervised voting. The supervised voting can also be considered as special case of remote voting where the problem of coersion is not a major concern. [1].

We wish to have a voting system where the voter never gets to contact the authorities and just casts his vote during the voting phase but such a system is nearly impossible to achieve as the authorities first need to establish the identity of the voter. In Civitas as well, the voter is required to contact the registration tellers in order to obtain his credentials.We assume that voter is able to do so without any effect from adversary [1].

Security Properties: Civitas is required to satisfy verifiability should protect voter privacy and the voting system should be coercion resistance. The final tally should be verifiably correct with respect to both vote verifiability and universal verifiability.

One of the ways to fulfill the confidentiality requirement is by guaranteeing anonymity, meaning that the system server never clearly reveals how a voter voted. But this is not sufficient for a remote voting system like Civitas. Voters can gain additional information during the voting phase which could enable them to sell their votes. Such information can also be used to coerce voters and in this case the coercer could be someone present at the time of voting like family member or employer or friend.

We use the concept of non-provability to provide coercion resistance. We make sure that a voter is not able to prove to the anyone whom he voted for. Thus even if adversary tries to coerce the voter, it knows that the voter can easily fool it and vote according to his wish.

The adversary model we use can perform many types of attacks. It is able to corrupt a threshold of the election authorities, demanding their secret keys,

controlling all public channels on the networks, corrupting a threshold of the voters, asking the voter to reveal his secret credential etc. Civitas is secure against all attacks from such adversaries.

2.2 Implementation of Civitas

Agents: To implement Civitas we need various types of agents. The voters are of course the very basic requirement of a voting system. The other agents required are supervisor, registrar, registration tellers and tabulation tellers [1]. The functions performed by each of them are:

- The Supervisor administers an election. This includes specifying the ballot design and the tellers and starting and stopping the election.
- The registrar decides who all eligible to vote.
- Registration tellers generate the credentials that voters use to cast their votes.
- Tabulation tellers tally votes and declares the result.

Election Setup: It starts with the posting of the electoral roll on an empty bulletin board by the registrar. In this way all eligible voters have been identified. After that the tabulation tellers generate public key ensuring that the decryption of the message under this key requires presence of all the tellers. Finally all the credentials are generated to authenticate the voters. The credentials are pair of public-private key values.

Voting Phase: This is where the real power of Civitas is clearly visible. Adversary may try as much as he wants but the voter can easily prevent from being coerced. This phase starts with voters registering and acquiring their private credentials. Each registration teller authenticates a voter using the voter's registration key and voter gets its credentials. Voter receives small part of credential from each teller and combines them to form the credential. For voting the voter has to submit a choice of the candidate and private credential in encrypted form. This submission does not require either of the voters' key. The vote is collected in all ballot boxes. This ensures that even

if any ballot box gets corrupt then also the vote will get counted in final tally.

Resisting Coercion: The key idea that enables the Civitas to resists coercion and defeats the selling of votes is that a vote can be cast using a fake credential. So if an adversary tries to coerce a voter, the voter may do as he wishes but by using fake credentials. The vote with fake credentials will be removed during tallying phase. Thus the voter can easily make the adversary a fool and cast his vote later. Also the voter is free to submit more than one vote in which case only the last vote will be counted.

To construct a fake credential, a voter chooses at least one registration teller and substitutes a random group element for the share that registration teller sent to the voter. The voter can construct a DVRP (Designated Verifier Re-encryption Proof is used to prove to the adversary that the fake credential is re-encryption of the public credential share) that causes fake share to appear real to the adversary.

Civitas is compatible with the use of any ballot design for which the proof of well formedness is possible.(citation)

2.3 TABULATION PHASE

The tabulation tellers tally the elections collectively. Firstly each teller retrieve the votes from each ballot box and also from the public bulletin board. Then they only keep the votes whose proof of well-formedness is available. In other words they discard all the ballots which are not well formed. Now since every voter has equal rights so all the duplicate ballots are removed. While removing the duplicate ballots, care is taken that only the last ballot remains and previous ones are removed. Then before removing the fake ballots they are anonymized so that it cannot be known which ones from the bulletin board were fake. The fake ballots are removed and then the result is declared along with a proof of correct counting.

Removing the duplicates and authorized votes would be easier if we had the power to decrypt but this pose a problem to coercion resistance and even anonymity. Because the voters private credentials becomes vulnerable. Instead a zero knowledge proof PET (plaintext equivalence test) is used to compare ciphertexts. This helps preserve voter privacy and thus supports coercion resistance.

The main components of the tabulation phase are the ballot boxes and the mix networks. The ballot boxes are log only services on which data

can be written only once and no data can be removed. They task is to report all their content after the end of the elections. At the end of the elections, each ballot box posts a commitment to all its content on the bulletin board. The supervisor posts his signature on all commitments and thus clearly defining all the votes that will be used in tallying phase. The second component is the mixnet. It plays a very important role in preserving the voter privacy. Before the fake ballots are removed, the ballots are anonymized so that adversary/voter cannot infer which ones from the bulletin board are fake. This anonymization is done by the mixnets. To reduce dependence on a single mixnet, multiple mixnets are used.

2.4 Verifying Elections

The tabulation tellers are required to post proof that they have done their assigned task correctly. And all tabulation tellers verify those proofs as the tabulation proceeds. Since we have already assumed that there is at least one honest teller, he refuses to continue as soon as he discovers anything wrong. These proofs are publicly verifiable and an external judge can also do the same. Thus we have ensured universal verifiability. Together with the ability of a voter being able to verify that his vote is present in the ballot box it ensures end-to-end verifiability in the absence of clash attacks.

2.5 Scalability

With the current implementation of Civitas, the duplicate removal method is brute force. Same is applicable for the fake removal method. In both cases we just compare each credential with each one in the list leading to the time complexity being quadratic. This poses some problems in the usage of Civitas in national level elections, limiting its usage to just small scale election like university elections. Certain methods have been proposed in the past to improve the time complexity like grouping the voters, but they haven't been useful. A lot of work is being done in the subject and improvements are expected in near future.

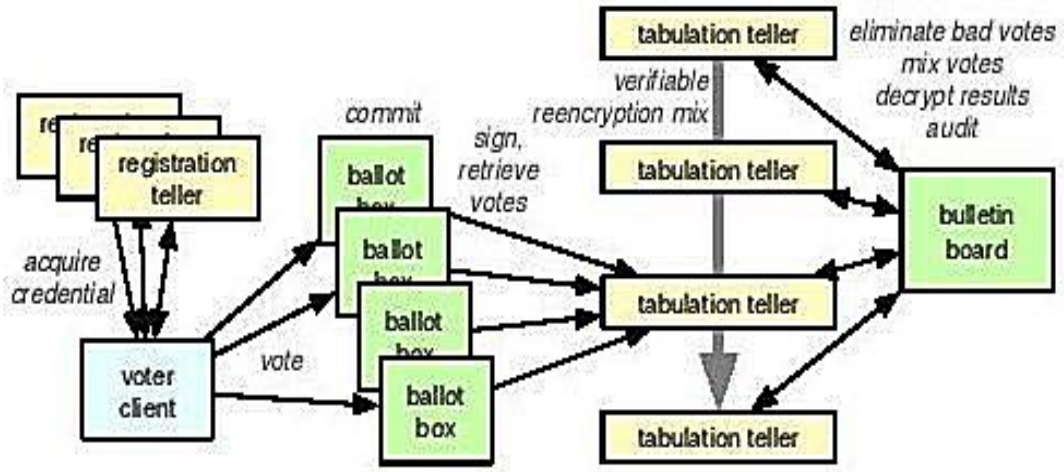

Figure 1: Civitas Architecture[1]

3 Time Complexity of Civitas

The implementation of civitas as explained above is $O(max(m^2, mn))$ where m is the number of votes cast. This makes the use civitas limited to just small scale elections. This is mainly because of two factors. The duplicate removal and the fake removal method. Both these methods have their implementation in brute force way.

In duplicate removal we check credential from each ballot against each each other ballot using PET proofs making it quadratic in m(number of votes cast). Similar in removing fake credential. We check each credential against each of the encrypted real credentials thus giving us O(mn) time complexity where n is the number of eligible voters. At both these places we have used brute force methods without using any properties.

4 Some Prior Zero knowledge Proofs

We present some prior zero knowledge proofs which we will be using later in our methods. We already have proofs for equality called Plaintext Equivalence Test(PET).

[1]This figure has been taken from "Civitas: Toward a Secure Voting System" [1]

Sorting a list: To present a proof of sorting. We take a list of ciphertexts $a_1, a_2...a_n$ and present a sorted version $a_{\pi(1)}, a_{\pi(2)}, ...a_{\pi(n)}$ where π is a permutation. Then we present n proofs, each on a pair $a_{\pi(i)}<a_{\pi(i+1)}$ where i ϵ {1,2...n-1}.

Stable Sorting: We already have a proof of sorting in which the entire process takes O(nlogn) time. We just modify a little bit to get a proof for stable sorting as well. We have a list $a_1, a_2...a_n$. To the i^{th} member in the list we concatenate i. After that we sort normally and $\pi\prime$ denotes the stable sorted list. Now we have got a list $a^j_{\pi\prime(1)}, a^k_{\pi\prime(2)}, ...a^z_{\pi\prime(n)}$ where the number on top is the one we concatenated in the last step. Now we again present n proofs, one for each consecutive pair $a^j_{\pi\prime(i)}, a^k_{\pi\prime(i+1)}$ either $a_{\pi\prime(i)}<a_{i+1}$ or $a_{\pi\prime(i)} = a_{\pi\prime(i+1)}\&j<k$. This becomes proof of stable sorting.

5 Changes in Civitas

Our main contribution to civitas the changes in the cryptography and the changes in the proofs that the authorities present to verify that they have worked correctly. Most of the time we have used the very basic algorithm like binary search and sorting algorithms. We assume that the reader is already familiar with their working. We implement some modified versions of these algorithms.

5.1 Changes in the cryptography

We use the same method as normal version for obtaining credential from authorities. The voter obtains a part of the credential from each authority. And then he combines them to obtain the complete secret credential. But we suggest a small change in the combination process. To form his credential the voter just concatenates all his credentials in a specific order which is publicly available. This method is publicly available so anybody knowing all the secret parts can form the credential. The authorities also make another method of permutation available using which another secret credential is made by the voter. Also each authority has its own private/public key pair. To send the data to the authority the data has to be encrypted with the public key of

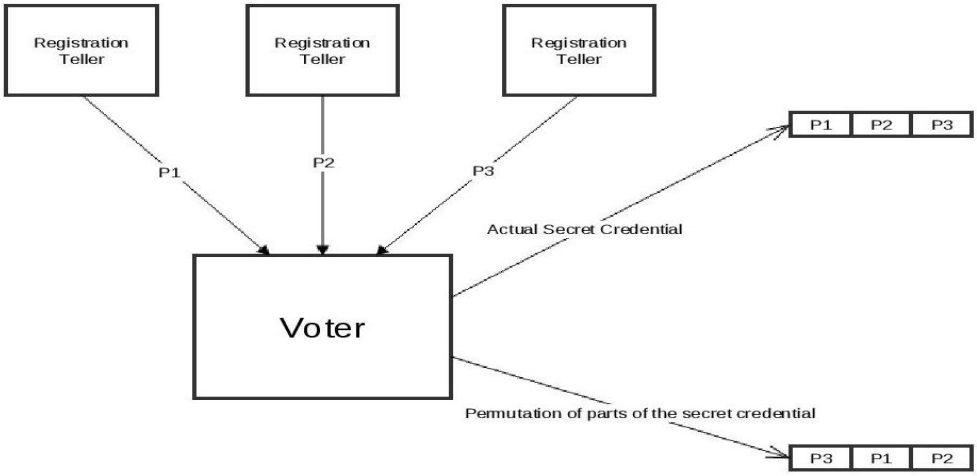

Figure 2: Formation of credential by voter

the authority. Also there is a joint key whose data can be decrypted by all authorities together.

Figure 2 shows the process of formation of both credential by the voter and Figure 3 shows contents of the ballot box when the voter sends it. The auxiliary data is other data which is optional and is different for different elections.

When the voter sends his ballot, he encryptes each part of the credential with corresponding key for both the permutation. The ballot box now contain two encrypted credentials and a encrypted vote which is encrypted with the joint key.

5.2 Removing Duplicates

For the decryption we give the entire list to the first authority. The authority makes a copy, decrypts the first portion of the credentials and sorts them according to this value. Also each member in the sorted copy has a pointer to its original value in the original list. The authority presents a proof of sorting. Now the authority starts marking the ballots. The first ballot is marked A11. Now if the second ballot has same value of the first part of the credential as first ballot then it is marked same else A12 and so on. similarly second authority marks them as Bxy. These markings are then transferred

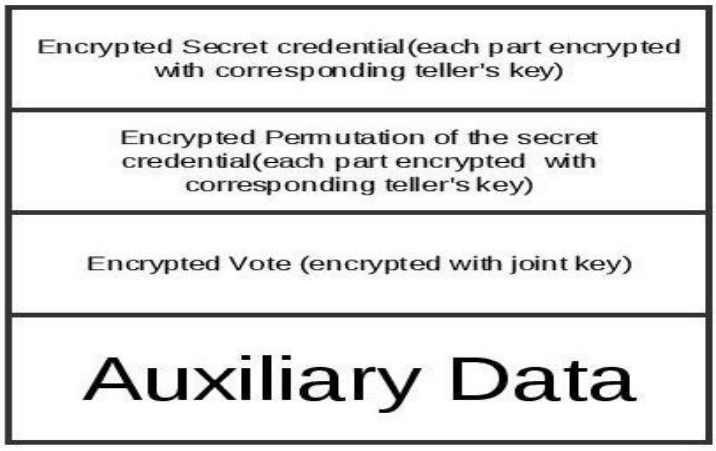

Figure 3: The content of the ballot

to the original list. This process is done by each authority. After all the authorities are finished marking we concatenate the markers and stable sort the list according to the strings formed from the concatenation. The proofs of stable sorting will also be O(m) where m is the number of voters. Now we know that same credential will occur consecutively so they can be removed easily.

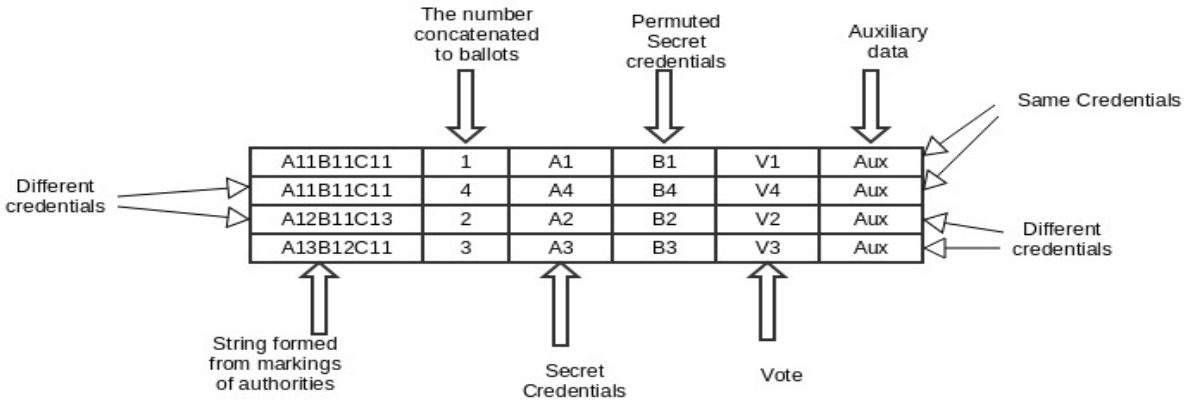

Figure 4: Removing duplicate ballots

Figure 4 depicts the final list which is obtained after all the authori-

ties have marked the ballots and it has been stable sorted on the basis of the strings so formed. The first and second ballots have same string "A11B11C11" so the are equal. Hence we will keep only the second ballot. The rest of the ballots are all different hence after removing the first ballot no duplicates are left.

In the entire process we present O(m) proofs and sorting takes at most O(mlogm) time. So we have finally removed all the duplicates in O(mlogm) time. At each step the corresponding proofs have been presented by the authorities in similar manner as given in Section 4.

5.3 Removing the Fake Credentials

Now the other permutation comes into play. We have list of ballots without duplicates. Each ballot contains two credentials. First of all the first credential is removed from each ballot as shown is Figure 5.

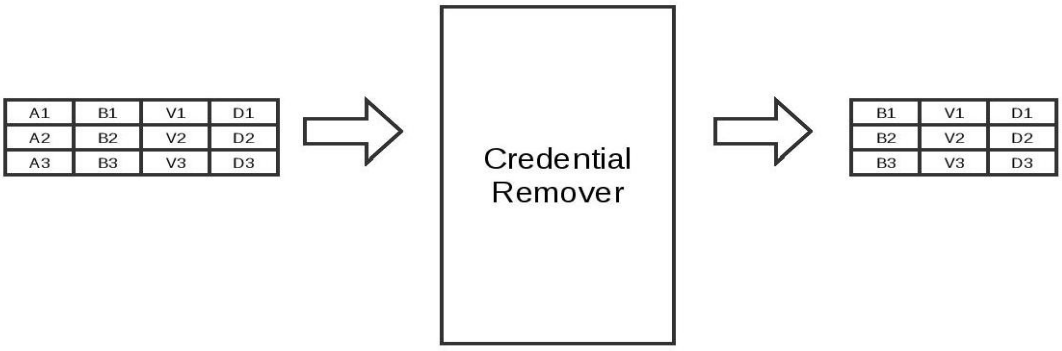

Figure 5: Removing the credentials from the ballots

Then this list is passed through mixnet which results in re-randomization and permutation of the list. We call this list X. We also take the list of permuted real credential (encrypted) and pass it through this mixnet. We call this list Y. The entire process is shown is Figure 6.

Now sort this list according to the decrypted values. For each value in the list Y do a binary search in this list. The binary search proceeds as follows:

- The first authority takes the ballot box decrypts, its portion of the credential and checks it with the corresponding portion of the middle credential in the list Y and also presents a proof of what is observed

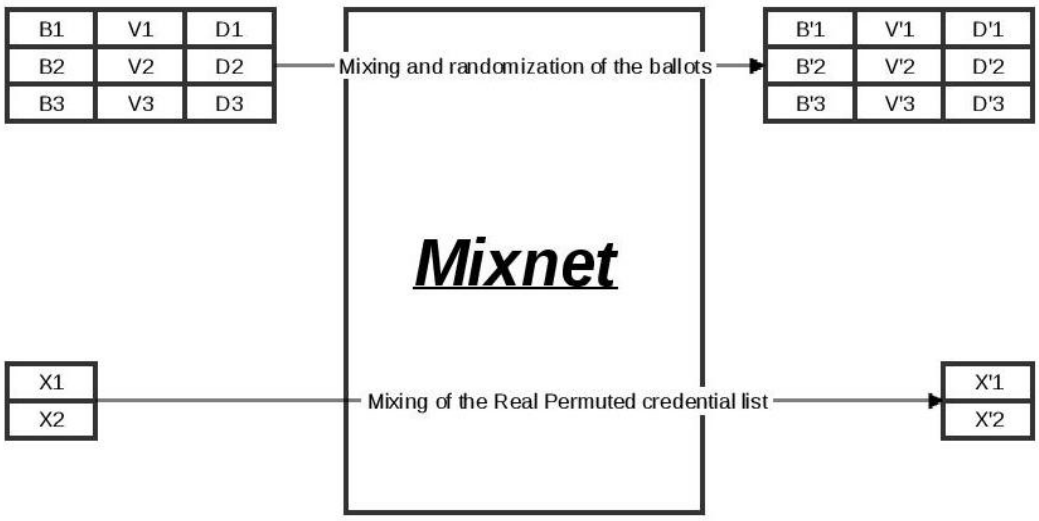

Figure 6: Mixing of ballots and real credential list

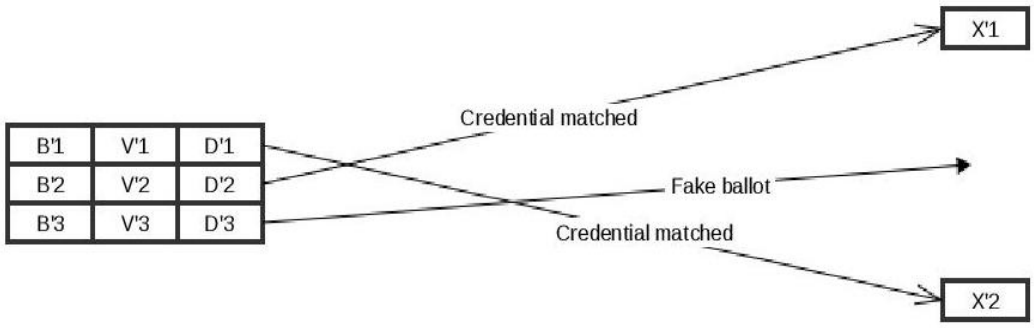

Figure 7: Identifying the fake ballots and removing them

since we have proofs for Plaintext equivalence and Less than ot Greater than relation. If it is lesser than that then the search space reduces to first half portion of the list. If it is more than that then the search space reduces to last half of the list. If it is equal then the entire credential is passed to the second authority which checks for the second portion of the credential with the middle element of list Y. The process continues till the ballot has been identified as authentic or real.

The final proof we get can be of two form. Either the credential is equivalent to something in the list (ballot corresponding to B'1 and B'2 in Figure 7), or it lies between two element of the list or it is outside the bounds of the list (ballot corresponding to B'3 in Figure 7). In any case atmost two proof suffice. The binary search takes at most k+logn time for a element where k is the number of authorities. Assuming k is small compared to logn the time complexity of the entire work is $O(mlogn)$. Thus all fake credentials have been removed in O(mlogn) time.

The final time complexity of the protocol is $O(max(mlogm, mlogn))$

6 Conclusion

We have explained the implementation of civitas e-voting protocol with emphasis on time complexity. After that we have suggested some changes in it which have been shown to bring down the time complexity from quadratic to nlogn and thus makes the protocol more usable in large scale elections. This can significantly change the election process and can serve as a great step towards electronification of election process.

References

[1] Andrew C. Myers Michael R. Clarkson, Stephen Chong. Civitas: Toward a secure voting system. *IEEE Symposium on Security and Privacy*, 2008.