Specifications of Electronic Voting Machine

Pranav Baurasia (2016CS10369)

May 26, 2018

Abstract

It is debatable if current direct-recording electronic voting machines can sufficiently be trusted for a use in elections. Reports about malfunctions and possible ways of manipulation have been in news in the last year. The issues have been raised in the parliament also. Voting schemes have to fulfill seemingly contradictory requirements: On one hand the election process should be verifiable to prevent electoral fraud and on the other hand each vote should be deniable to avoid coercion and vote buying. In this paper, I considered some of the problems, if not all, that are present in the current election system in India. I identified poor practices in the commissioning, development operation and scrutiny of the system and then present the suggestions for improving the system.

1 Introduction

Fundamental right to vote in elections forms the basis of democracy. All earlier elections be it state elections or centre elections a voter used to cast his/her favorite candidate by putting the stamp against his/her name and then folding the ballot paper as per a prescribed method before putting it in the Ballot Box. This is a long, time-consuming process and very much prone to errors. But this has been changed completely by the EVMs. Electronic Voting Machine (EVM) is a simple electronic device used to record votes in place of ballot papers and boxes which were used earlier in conventional voting system. No more ballot paper, ballot boxes, stamping, etc. all this condensed into a simple box called ballot unit of the electronic voting machine. But in the recent time, various challenges have been made against tampering of the current EVM so in this document I am giving a design which will try to rectify the various problems present in the given EVM. This design is coercion resistant so that no forces can change the vote of the voters and secure enough such that it can be trusted by all and gives the anonymity to voters as well.

2 What should be the specifications of EVM?

One question that arises when designing an EVM is what specifications should it have? This is a very debatable question and various requirements have been posed for the EVMs and many requirements are generating day by day. But there are some minimum requirements which must be fulfilled by the EVMs. I have tried to address these issues.
The specifications or features I have found which should be there in the EVMs are the following :
-

1. **Coercion freedom :-**
   Coercion is the action or practice of persuading someone to do something by using force or threats. This is the problem that has the highest priority. The coercer should not be able to identity to which candidate the voter has given his vote.
2. **Secrecy :-**
   Secrecy is something that should be there in any voting system. To whom the voter has given vote should not come out in any case. The voter has the right to keep his vote secret.
3. **Non-repudiation :-**
   It means I don't know to whom you voted but I can assure you that I counted correctly. This feature enables the voter to trust the EVM and voting system.

4. **Verifiable :-**
   Another feature is verifiability. The EVM should be able to be checked and demonstrated to be accurate. This gives EVM the power of authenticity.

5. **Crypto-secure :-**
   All the technology used should be 100 percent secure. Then only the voting machines are reliables otherwise there is no need of EVMs. Then balloting will be better in any case.

6. **Auditable :-**
   EVM should be able to be (if asked) prove that no biasing occurs in it. This feature also increases the credibility of voting system.

7. **Anonymity :-**
   Anonymity of the voter is the biggest concern of them because they can be in threat by the party leaders or some "Gundas" and if they know their vote is not anonymous then will not vote for the deserving candidate (I mean the candidate whom they think are deserving) and will severely affect the voting system.

8. **Self-certifiability of the EVM** :-
   This property which I think is the most important in relation to India. Present EVMs have tampering problem but another measures are taken for it but it should be the property of the machine that it is tamper proof in terms of both hardware and software.

9. **Information leakage :-**
   One of the biggest concerns about the EVMs are that they should not leak the data of the voters and to whom they vote. No leaking of data should be possible in the EVM. It is the design decision taken in the process of designing which will ensure data leakages such as using cryptographic security measures to store the votes.

10. **No communication Interfaces :-**
    Another concern about the system is that it should not have any communication interface which can provide any form of access because if access is there then the security risks itself increases.

11. **Entity of individual piece :-**
    Current EVMs have many parts and are assembled. Any extra device can be added or tampering can be done during the assembling case and this will be very hard to detect. The software should be encoded into hardware and the hardware should be a single entity so if something happens Iwill be sure that it is not from the manufacturer's side.

12. **Hardware encrypted :-**
    The hardware encryption should be used to replace software, in the process of data encryption.

13. **Secret device:-**
    All the source code of hardware, software and firmware should be kept secret because if someone gets it then he/she will get the hands on it and it will increases the chances of getting EVMs hacked or tampered.

3 System Design

EVM will consist of three components, namely:-

- **Control Unit**
  The Control Unit is the control section of the EVM and controls the polling process. This is the heart of the EVM where the all votes will be stored and later will display the results of elections after the polling process. It is a single piece device with a single circuitry and one-time assembled body what I mean by one-time assembled is that if opened after assembling then whole device will break. This will ensure no tampering will happen.

- **Balloting Unit**
  The Balloting Unit of EVM is a small Box-like device, on top of which each candidate and his/her election symbol is listed like a big ballot paper. Against each candidate's name, a red LED and a blue button is provided. The voter polls his vote by pressing the blue button against the name of his desired candidate. This also has one-time assembled body.

- **VVPAT**
  Voter Verifiable Paper Audit Trail (VVPAT) or Verifiable Paper Record (VPR) is a method of providing feedback to voters using a ballot-less voting system. A VVPAT is an independent verification system for voting machines designed to allow voters to verify that their vote was cast correctly, to detect possible election fraud or malfunction, and to provide a means to audit the stored electronic results.It contains name of the candidate (for whom vote has been casted) and symbol of the party/ individual candidate. But one feature which differentiates it with the current one will be that it will not give output of the button I pressed but it will decrypt the information stored and return the result of the output of the El Gamal algorithm.

4 Hardware integrity:

Current EVMs has various components which are built all over the world and finally come to India and then gets assembled. This makes the process of manufacturing EVMs long and has high security threats because anything can be put in the process or some elements of circuits can changed by similar circuits which can be biased for any candidate. Also various authorities will come into question if some tampering during the manufacturing or assembling of EVMs comes out. Iwill be not be able to claim that one person or organization has done it. So, the hardware used for making EVMs should be a single device with no components coming from different companies. Also, once assembled and circuit put into a box it can be opened without breaking the box. This will surely ensure that if tampering happens, clear signs of scratches or breaking of device can be seen.

5 Mathematical Model of EVM

An election system consists of several sets of entities:

1. Registrars: Denoted by $R = \{R_1, R_2, ..., R_{n_R}\}$,this is a set of n_R entities responsible for jointly issuing keying material, i.e., credentials to voters.

2. Authorities (Talliers): Denoted by $T = \{T_1, T_2, ..., T_{n_T}\}$, authorities are responsible for processing ballots and jointly counting votes and publishing a final tally.

3. Voters: The set of $\{n_V\}$ voters, denoted by $V = \{V_1, V_2, ..., V_{n_V}\}$, are the entities participating in a given election administered by R. Ilet i be a public identifier for V_i.

Imake use of a bulletin board, denoted by BB. This is a piece of universally accessible memory to which all players have appendive-write access. In other words, any player can write data to BB, but cannot overwrite or erase existing data. Moreover, voters will be able to read the contents of BB once the vote casting phase has ended. For notational convenience, Iassume that data are written to BB in μ-bit blocks for an appropriate choice of μ. Shorter data segments may be padded appropriately. For simplicity of exposition, Iassume no ordering on the contents of BB.

6 Functioning of EVM

Idefine a candidate slate C to be an ordered set of n_C distinct identifiers $\{c_1, c_2, ..., c_{n_C}\}$, each of which corresponds to a voter choice, typically a candidate or party name. In an election, choice c_j may be identified according to its index j. Thus, for cryptographic purposes the candidate slate consists of the integers $(1, 2, ..., n_C)$ and may be specified by n_C alone. Idefine a tally on an election under slate C to be a vector X of n_C positive integers $x_1, x_2, ..., x_{nC}$ such that x_j indicates the number of votes cast for choice c_j . The protocols composing an election system are then as follows:

1. **Registering:** The function register $(SK_R, i, k_1) \rightarrow (sk_i, pki)$ takes as input the private registrar key SK_R, a (voter) identifier i and a security parameter k_1, and outputs a key pair (sk_i, pk_i). This is computed jointly by players in R, possibly in interaction with voter V_i.

2. **Voting:** The function vote $(sk, PK_T, n_C, \beta, k_2) \rightarrow$ ballot takes as input a private voting key, the public key of the authorities T , the candidate-slate specification n_C, a candidate selection β, and a security parameter k_2, and yields a ballot of bit length at most μ. The form of the ballot will vary depending on the design of the election system, but is in essence a digitally signed vote choice encrypted under PK_T .

3. **Tallying:** The function tally $(SK_T, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3) \rightarrow (X, P)$ takes as input the private key of the authority T , the full contents of the bulletin board, the candidate-slate size, all public voting keys, and a security parameter k_3 and outputs a vote tally X, along with a non-interactive proof P that the tally was correctly computed.

4. **Verifying:** The function verify $(PK_T, BB, n_C, X, P) \rightarrow \{0, 1\}$ takes as input the public key of the authorities, the contents of the bulletin board, the candidate-slate size, the voting tally, and a non-interactive proof of correct tallying. It outputs a '0' if the tally is incorrect and a '1' otherwise.

Idefine an election scheme ES as the collection of these functions. Thus ES = {register, vote,tally, verify}//

Design decision: There are many election models in use throughout the world. The model Ipropose here excludes important variants. In some systems, for example, voters are asked to rank candidate choices, rather than just listing those they favor. Many systems permit the use of write-in votes, i.e., the casting of a ballot in favor of a candidate not listed on the slate for the election. Iexclude write-in voting from my model because it undermines the possibility of coercion resistance in any scheme where an observer can see a complete election tally including write-in votes. An attacker may, for example, require coerced voters to cast write-in ballots for candidate names consisting of random strings pre-specified by the attacker. This way, the attacker can: (1) Verify that coerced voters complied with instructions, by looking for the random strings the attacker furnished, and (2) Ensure that the votes of coerced voters are not counted, since random strings will most likely not correspond to real election choices. (Thus, this would combine the forced abstention attack.)//

7 Assumptions

1. **Assumptions in setup phase:** my security definitions permit the possibility of static, active corruption by the adversary of a minority of players in T in the setup phase. The security of my construction then relies on generation of the key pair (SK_T, PK_T) by a trusted third party, or, alternatively, on an interactive, computationally secure key-generation protocol between the players in T.

2. **Assumptions prior to registration:** The adversary cannot coerce a voter prior to the registration phase in the sense of requesting in advance that the voter retain transcripts of the registration process, or by providing data in an attempt to dictate voter interaction with the registrar.

3. **Assumptions in registration phase:** Iassume that this phase is trustworthy. Strong integrity of the registrar R is of course critical for any secure election system. To evade coercion, a voter must be able to receive a credential without adversarial interference. An adversary capable of corrupting and seizing the credentials of a voter in this initial phase can attack the whole system. An adversary capable of preventing a voter from registering can mount a forced-abstention attack. Iassume therefore that the voter receives her credential from R via an untappable channel. Iare helped here by the fact that registration is generally an off-line procedure. For example, a voter might receive his/her registration through the postal service. Imust then assume that mail in the postal system is not subject to compromise. Alternatively, registration could be performed by voters in person.

4. **Assumptions on voting, tallying and verification phases:** After the registration phase, Iassume that the adversary will not seize control of a minority of players in T and any number of voters.

8 Election Protocol

This section describes the protocol for security used in this system. It also contains the description of the cryptographic building blocks used.

8.1 Threshold cryptosystem with re-encryption:

My first building block is a threshold public-key cryptosystem CS that permits re-encryption of ciphertexts with knowledge only of public parameters and keys. The private key for CS is held by T in my construction. To describe my aim in the ideal, I would like any ciphertext E to be perfectly hiding. I would like decryption to be possible only by having a majority of players in T agree on a ciphertext to be decrypted. I model this latter ideal property as in terms of a special decryption oracle denoted by DEC . I assume further that any decryption performed by DEC is publicly verifiable.

8.2 Selected cryptosystem:

The El Gamal encryption seems a natural choice of cryptosystem for my purposes. The system provides an additional layer of security by asymmetrically encrypting keys previously used for symmetric message encryption.

8.3 Plaintext Equivalence Test (PET):

A plaintext equivalence test (PET) is cryptographic primitive that operates on ciphertexts in a threshold cryptosystem. The input to PET is a pair of ciphertexts; the output is a single bit indicating whether the corresponding plaintexts are equal or not. PET may be realized as an efficient distributed protocol that reveals no additional, non-negligible information about plaintexts.

8.4 Mix network:

A (re-encryption) mix network (MN) is a distributed protocol that takes as input an ordered set $E = \{E1, E2, ..., E_d\}$ of ciphertexts generated in a cryptosystem like El Gamal that permits re-encryption. The output of MN is an ordered set $E' = \{E'_{\pi(1)}, E'_{\pi(2)}, ..., E'_{\pi(d)}\}$. Here, $E'_{\pi(i)}$ is a re-encryption of E_i, while π is a uniformly random, secret permutation. This is to say that MN randomly and secretly permutes and re-encrypts inputs. Thus, the special privacy property of a mix network is this: An adversary cannot determine which output ciphertext corresponds to which input ciphertext, i.e., which inputs and outputs have common plaintexts. Stated another way, an adversary cannot determine $\pi(j)$ for any j with probability non-negligibly better than a random guess. A number of good mix network constructions have been proposed that offer privacy and robustness against a static, active adversary capable of corrupting any minority of the n players (servers) performing the mix network operation. These constructions can offer the additional property of verifiability. In other words, a proof is output that is checkable by any party and demonstrates, relative to E and the public key of the ciphertexts that E is correctly constructed. It is convenient to conceptualize MN as an ideal primitive with the property of public verifiability.

9 Proposed Protocol for Elections

- **Setup :-**
  The key pairs (SK_R, PK_R) and (SK_T, PK_T) are generated (in an appropriately trustworthy manner, as described above), and PK_T and PK_R are published along with all system parameters.

- **Registration:-**
  Upon proof of eligibility from V_i, the registrar R generates and transmits to V_i a random string $\sigma_i \; \epsilon_U \; G$ that serves as the credential of the voter. R then adds $S_i = E_{PK_T}[\sigma_i]$ to the voter roll L. The voter roll L is maintained on the bulletin board BB and digitally signed as appropriate by R.

- **Candidate-slate publication:**
  R or some other appropriate authority publishes a candidate slate C containing the names and unique identifiers in G for n_C candidates, with appropriate integrity protection. This authority also publishes a unique, random election identifier ϵ.

- **Voting:**
  Voter V_i casts a ballot for candidate c_j comprising M-El Gamal ciphertexts $(E_1^{(i)}, E_2^{(i)})$ respectively on choice c_j and credential σ_i. In particular, for $a_1, a_2 \epsilon_U Z_q$:
  $E_1^{(i)} = (\alpha_1, \alpha_1^{'}, \beta_1) = (g_1^{a_1}, g_2^{a_1}, c_j h^{a_1}), E_2^{(i)} = (\alpha_2, \alpha_2^{'}, \beta_2) = (g_1^{a_2}, g_2^{a_2}, \sigma_i h^{a_2})$.
  The first is a cipher-text on the candidate choice of the voter, the second a cipher-text on the credential of the voter. Additionally, V_i includes NIZK proofs of knowledge of σ_i and c_j, and also a NIZK proof that $cj \epsilon C$, i.e., that c_j represents a valid candidate choice. The latter can be accomplished, for example, using a disjuctive proof that the ciphertext constitutes a valid encryption of a candidate choice in C. It is needed because an invalid candidate choice is like a write-in: It can effectively serve as a receipt. These NIZK proofs, which I denote collectively by Pf, may be accomplished efficiently using standard techniques. As is standard practice, the challenge values for Pf are constructed using a call to a cryptographic hash function. Input for these challenge values includes , E1, E2 and commitment values required for realization of the NIZK proofs. V_i posts $B_i = (E_1, E_2, Pf)$ to BB via an anonymous channel.
  Note: This is not a receipt! By my definitions, the adversary may not know whether or not a given voter has even posted a vote. The adversary therefore can't simply coerce by requesting decryption information.

- **Tallying:** To tally the ballots posted to BB, the authority T performs the following steps:

  1. **Checking proofs:** T verifies the correctness of all proofs on BB. Any ballots with invalid proofs are discarded. For the valid, remaining ballots, let A_1 denote the list of ciphertexts on candidate choices (i.e., the E_1 ciphertexts), and let B_1 denote the list of ciphertexts on credentials (i.e., the E_2 ciphertexts).
  2. **Eliminating duplicates:** The tallying authority T performs pairwise PETs on all ciphertexts in B_1, and removes duplicates according to some pre-determined policy, using e.g., order of postings to BB. When an element is removed from B_1, the corresponding element (i.e., that with the same index) is removed from A_1. I let B_1' and A_1' be the resulting "weeded" vectors. This is equivalent to retaining at most one ballot per given credential.
  3. **Mixing:** T applies MN to A_1' and B_1' (using the same, secret permutation for both). Let A_2 and B_2 be the resulting lists of ciphertexts.
  4. **Checking credentials:** T applies mix network MN to the encrypted list L of credentials from the voter roll. T then compares each ciphertext of B_2 to the ciphertexts of L using PET. T retains a vector A_3 of all ciphertexts of A2 for which the corresponding elements of B2 match an element of L according to PET. This step achieves the weeding of ballots based on invalid voter credentials.
  5. **Tallying:** T decrypts all ciphertexts in A_3 and tallies the final result.

10 Correctness of model

10.1 Correctness:

I first consider the property of correctness. This property is in fact twofold: First, it stipulates that an adversary A cannot pre-empt, alter, or cancel the votes of honest, i.e., voters that are not

controlled; Second, it stipulates that A cannot cause voters to cast ballots in such a way as to achieve double voting, i.e., use of one credential to vote multiple times, where more than one vote per credential is counted in the tally.
In my experiment characterizing correctness, I give the adversary powers she does not normally have. Namely, apart from getting to select a set V of voters she will control, I also allow her to choose the candidate-slate size n_C, and to choose what votes will be cast by voters she does not control. The latter voters will indeed vote according to the adversary's wish – but only for the purposes of my thought experiment defining correctness, of course. If the adversary still cannot cause an incorrect tally to be computed (i.e., one not corresponding to the votes cast), then the scheme has the correctness property even in the real-world scenario in which the adversary has less power. The aim of the adversary is to cause more than $|V|$ ballots to be counted in the final tally on behalf of the controlled voters, or to alter or delete the vote of at least one honest voter. (This corresponds to the the condition that: (1) The verification of the tally succeeds, and (2) That either a vote is "dropped" or "added".) my definition assumes implicitly that tally is computed correctly by the authority T . (The next property I consider, namely verifiability, addresses the possibility that this is not so.) In what follows, I let $< Y >$ denote the multiset corresponding to entries in the vector Y , and $|Y|$ denote the cardinality of set Y.

10.2 Verfiability:

As explained above, an election system has the property of correctness if computation of tally always yields a valid tabulation of ballots. Given the ability of an adversary A, however, to corrupt some number of authorities among T , I cannot be assured that tally is always computed correctly. The property of verifiability is the ability for any player to check whether the tally X has been correctly computed, that is, to detect any misbehavior by T in applying the function tally. A strong security definition for verifiability is appropriate given the high level of auditability required for trustworthy elections. Such a definition considers an attacker A capable of controlling all of the voters and tallying authorities in T . This attacker seeks to construct a set of ballots on BB and a corresponding tally X and proof P of correct tabulation such that the proof is accepted by verify, but the tally is in fact incorrect. By an incorrect tally, I mean one in which all of the valid ballots of a particular voter (i.e., corresponding to a particular credential) are discounted, or else where multiple votes are tallied that could have been generated by the same voting credential.

Another aspect of verifiability that I do not formally define, but do mention here and incorporate into our proposed protocol is that of verification against voter rolls. In particular, it may be desirable for any election observer to check that credentials were assigned only to voters whose names are on a published roll. This is not technically a requirement if I rule out corruption of players R, but may still be desirable for high assurance of election integrity.

11 Conclusion

EVMs described in this paper provides a secure and coercion free voting system. I have tackled the issues of anonymity, coercion, security, correctness, verifiability of EVMs, secrecy, non-repudiation, crypto-secure, auditable, self-certifiability. As I mentioned, no communication interfaces have been attached to the EVMs. Information leakage if happens also then due to the El Gamal algorithm the data will be useless because it has to be decrypted which cannot be done without the keys. Entity of individual piece is also determined by using single circuitry and one time assembled body. How one use cryptographic security algorithms in the EVMs - this is what I have tried to explain in this paper. This type of EVM will not only gain the trust of people but also serves its purpose. EVMs are designing the fate of our country and it is very important that it is trustworthy and tamper proof.

12 References

- Coercion-Resistant Electronic Elections by Ari Juels and Dario Catalano and Markus Jakobsson

- E-Voting and Identity by Aggelos Kiayias and Helger Lipmaa (Eds.)
- A. Shamir. How to share a secret. Communications of the Association for Computing Machinery , 22(11):612–613, November 1979.
- Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In Workshop on Practice and Theory in Public Key Cryptography (PKC '98). Springer, 1998.
- T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory , 31:469–472, 1985.
- P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In M. Yung, editor, CRYPTO '02, pages 385–400, 2002. LNCS no. 2442.
- M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, editor, Advances in Cryptology - Asiacrypt '00, pages 162–177. Springer-Verlag, 2000. LNCS No. 1976.
- Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. SIAM J. Comput., 20(6):1084–1118, 1991.
- J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In J. Kilian, editor, CRYPTO '01, volume 2139 of Lecture Notes in Computer Science, pages 368–387. Springer-Verlag, 2001.
- R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. The (in)security of distributed key generation in dlog-based cryptosystems. In J. Stern, editor, EUROCRYPT '99, pages 295–310. Springer-Verlag, 1999. LNCS no. 1592.
- V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In J. Pieprzyk and R. Safavi-Naini, editors, ASIACRYPT '94, pages 164–170. Springer-Verlag, 1994. LNCS no. 917
- B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In M. Wiener, editor, CRYPTO '99, pages 148–164. Springer-Verlag, 1999. LNCS no. 1666.