# A Three-Ballot Based Secure Electronic Voting System

Regivaldo G. Costa, Altair O. Santin, Carlos A. Maziero
Pontifical Catholic University of Paraná, Graduate Program in Computer Science
Curitiba – PR – Brazil
{rcosta, santin, maziero}@ppgia.pucpr.br

**Abstract** — This paper presents a secure Electronic Voting System, which provides the requirements and properties needed in voting environments, integrated in a single architecture. Our architecture mainly addresses the vote receipts, the uniqueness and materialization of the vote, and the privacy and anonymity of the voters. The architecture is based on the scattering of the responsibilities involved in the voting process among distinct interacting entities, to avoid possible critical security points. This proposal applies cryptographic techniques to achieve its security requirements. A proof-of-concept prototype was built, using web services and the Election Markup Language (EML), to show the viability of the proposal.

**Index Terms** — Electronic Voting System, Cryptography Based Security, Three-Ballot voting system

## INTRODUCTION

Electronic Voting Systems (EVS) are being increasingly used, replacing traditional paper-based systems. This tendency raises several security issues that should be considered, as the democracy principles depend on the integrity of the electoral processes.

The most relevant requirements of a secure EVS are the anonymity preservation, ensuring that a vote cannot be associated to its voter, and the voter's confidence on the correct accounting of her vote, through the generation of evidences or receipts of the vote assignment. Other properties are also expected from an EVS: to prevent vote trading and voter coercion; to produce trustworthy information about the voting procedures as a whole; to use homologated and certified software and hardware; materialization of the vote after assigned to a candidate – to make feasible the manual recount in the case of contestation of election results; to produce material evidences of each vote, to allow manual recount the votes in case of election appeals; and to use strong identification mechanisms, preventing voters impersonation.

Today, there is a wide understanding that the traditional voting systems should be computerized, for reasons like to reduce the vote counting time, to provide evidences that a vote is being correctly accounted, to reduce frauds (as fake voters), to remove errors in the ballot filling, and to improve the system usability, mainly for people with special needs [1]. We are conscious of concerns about the insecurity of software in general. Security incidents are very frequent, in many kinds of software and application domains. Therefore, at same time that some people defend the full computerization of the voting systems, others assume an opposite position.

Providing security to computer voting systems is not a trivial task. Beyond the classic security properties (integrity, confidentiality, and availability), other properties should also be ensured. There are some EVS requirements that seem contradictory, like: to ensure voter's authenticity and at the same time vote anonymity; to provide a vote counting proof, while preventing vote trade; to allow voting by Internet, but avoiding voter coercion; to guarantee the uniqueness of the vote in an decentralized voting system (aiming voter's anonymity and avoiding frauds), to allow voting automation while providing vote materialization (to allow recounting); and to ensure auditability in a software/hardware environment that may malfunction (due to malicious or unintentional actions).

Already proposed systems use complex mechanisms to ensure some EVS security requirements, like using visual cryptography to provide voting receipts [2], using a shared key to decrypt a vote using homomorphic encryption [3], using mix networks to create anonymous channels to ensure anonymity for the voter and the vote [4], among others. Alternatively, our proposal is based on classic cryptography techniques [5], using the standard public key cryptosystem and scattering the entities and separating their responsibilities, to avoid critical security points.

The proposal presented here aims go beyond the classic security properties by considering voting receipts, voter coercion, vote trade, vote materialization, voting process auditability, and voter anonymity and authenticity. A proof-of-concept prototype was built, using web services and the Election Markup Language (EML) to show the viability of the proposal. EML is a proposed standard for election data [6]. Web services were used to provide a standard for secure and interoperable system deployment. However, we do not assume that the proposal is an Internet based voting system.

## REQUIREMENTS FOR A VOTING SYSTEM

Each country defines a set of specific laws to rule its voting system, in order to establish its organization and to ensure its impartiality, integrity, and the democracy principles. Elections based in an Electronic Voting System (EVS) must comply with the laws and rules for voting systems (as discussed in [5,7,8]) and also fulfill the following main requirements:

- *Confidentiality*: the vote should be kept confidential from its verification and confirmation by the voter until the counting phase. Also, partial counting should not be possible.
- *Integrity*: the final vote counting must exactly represent the number of voters (vote uniqueness) and their intents (quality of the vote).
- *Availability*: an EVS should be dependable to guarantee the voting service availability and the respect to its security requirements during the entire election process.
- *Authenticity*: the voter authenticity must be verified at two distinct phases: at the voter registration, and just before the voting procedure. Voter impersonation should be prevented.
- *Anonymity*: the vote must remain anonymous during the entire voting process and after it; there should be no means to associate a vote to its voter, or vice-versa.
- *Vote receipts*: in an EVS, vote counting is done computationally and not under scrutiny by electoral authorities and society monitoring. A vote receipt should allow the voter to check if her vote was correctly counted, without allowing practices like voter coercion and vote trading.
- *No vote trading*: no voter should have access to material evidences that certify to other people the quality of her vote.
- *No voter coercion*: no party should have means to impose a voter to vote against her intents.
- *Uniqueness*: the voter should be able to vote just once in the same election.
- *Vote materialization*: the EVS must reproduce materially the quality of each vote, allowing manual vote recounting, if requested.
- *Auditability*: a voting system must provide audit trail of the entire voting process, for detecting frauds, software/hardware malfunction, or human operation errors. However, such information cannot keep, in no hypothesis, information that compromises the other security requirements.
- *Usability*: an EVS should be user-friendly, offering visual, touch, and audio resources that allow the voter to vote quickly and with no help from others.

The next section presents our proposal, which fulfills such properties.

## THE PROPOSED ARCHITECTURE

As fully electronic voting systems are not yet mature, new paper-based systems are still being proposed. Such systems introduce properties not present in conventional ones, like vote receipts.

Our architecture is fully computerized, but adopts the three-ballot scheme from the paper-based voting system proposed in [9]. That scheme uses three equal ballots for each vote, each one having a unique numeric identifier. The voter checks off her candidates in two ballots; for all the other candidates, she checks them off just once, on one of the three ballots, randomly. This way, the candidates she voted for will have two marks in the three ballots set, while all the other candidates will have just one mark each. Subsequently, one ballot chosen at random by the voter is copied for her as a vote receipt. Then, the three ballots are stored. After the election, all ballots copied as receipts are published, to allow voters to verify if their votes were taken into account.

Figure 1 presents an overview of our proposed architecture. It is built using the following entities: a Registration Agent, a Voting Console, a Voting Manager, an Electronic Ballot Box, and an Electronic Election Bulletin Board.

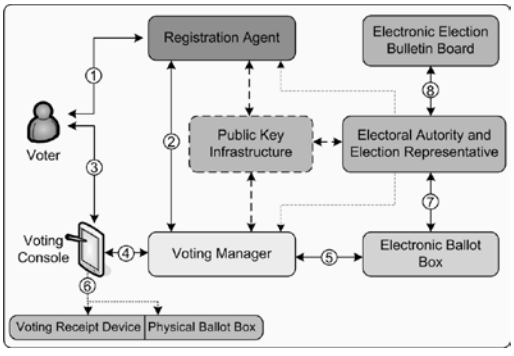


**Figure 1. Overview of the Proposed Architecture**

In order to take part in the voting process, firstly, the voter presents herself to the Registration Agent, to get a credential that qualifies her to vote (event 1 in figure 1). The Registration Agent interacts with the Voting Manager to obtain the corresponding Ballot IDs (event 2), and uses them to build the credential returned to the voter. Later on, after the authentication (event 3) the voter uses the Voting Console to vote (event 4) and the Voting Man-

ager stores the vote in the Electronic Ballot Box (event 5), while the Voting Console gives a voting receipt back to the voter (event 6). When the election finishes, the Electoral Authority and the Election Representatives start the counting phase (event 7), being votes counting and the receipts published in the Electronic Election Bulletin Board (event 8).

The architecture considers as actors the Voter, the Election Representatives and an Electoral Authority. In a general election, Election Representatives can be persons from the civil society and political parties, which are responsible for monitoring the voting process. The Electoral Authority manages the whole electoral process and enforces the voting rules and laws.

The voting process consists of three phases: the voter registration, the voting itself, and the storage and counting of votes. They are detailed in the following.

### The Registration Phase

The Registration Agent is the entity responsible for voters' admission and qualification during the registration phase, depicted in Figure 2. Its tasks include receiving voters at the polling station and requesting their identifications (either by biometry or another mechanism), to verify if they are able to vote. If so, voters receive credentials that enable them to the next phase (voting).

During its initialization (boot), the Registration Agent starts a Voters/Ballots ID Repository, using data from a repository of voters maintained by the Electoral Authority. Also, the Registration Agent requests $b$ Ballot IDs (*BID*) from the Voting Manager (events B1 and B2) and stores them locally in the Voters/Ballots ID Repository (event B4); $b$ can be defined by each Electoral Authority. The Voting Manager logs the *BID*s supplied to the Registration Agent in a local repository for Ballot IDs and Ballots (*BIR*), event B3. This initialization procedure makes unpredictable (for the Voting Manager) the sequence of voters accessing the Registration Agent, in order to prevent voter anonymity violations.

Once the registration phase starts, a voter should identify herself to the Registration Agent (event 1, figure 2). The Registration Agent verifies if the voter is able to vote (event 2), querying the Voters/Ballots ID Repository (*VBR)*. If so, it takes three random Ballot IDs out of the $b$ Ballot IDs present in *VBR*, signs them (compounding a credential) and returns them back to the voter (event 3). At same time, it updates the repository of voters (event 4) to register that the voter was qualified to vote, in order to assure vote uniqueness.

In order to keep $b$ Ballot IDs (*BID*s) in its Voters/Ballots ID Repository, the Registration Agent requires three new *BID*s to the Voting Manager (event 5). The Voting Manager chooses three new *BID*s (event 6), ciphers each one separately using the Voting Console's public key, sends them back to the Registration Agent (event 7*)*, and logs the *BID*s in the local repository for Ballot IDs and Ballots (event 8).

If the voter uses biometric authentication (event 1 in figure 2) to authenticate against the Registration Agent, a template of the voter's fingerprint is extracted, ciphered using Voting Console's public key and then attached to the credential (event 3). Such scheme guarantees the voter's authenticity and prevents frauds related to impersonation during the voting phase.

The random Ballot IDs sent by Registration Agent in event 3 (figure 2) are composed by three IDs that will be used by the voter during the entire voting process. They are not known by the Registration Agent, because Ballot IDs (*BIDs*) are ciphered using the Voting Console's public key; therefore, the Registration Agent performs a blind signature [10,11] on the *BIDs* composing the credential .

Interactions with the Public Key Infrastructure (events I and II) include procedures for signature authenticity verification, since all the transactions between entities in the entire process are digitally signed.

### The Voting Phase

The Voting Console is responsible for interacting with the Voter during the voting phase, depicted in Figure 3. Therefore, it is assumed that all messages from the Voting Console to the Voting Manager are resulting from interactions between the voter and the Voting Console.

If biometric authentication was adopted during the voter registration, the Voting Console gets the voter's biometric template, decrypts it and verifies its authenticity (event I, figure 3), through Registration Agent's digital signature. Then, the Voting Console requests voter's fingerprint, using a Biometric Device (*BD*). The biometric verification consists on comparing the template obtained from *BD* with the template coming from the Registration Agent. It is important to observe that no information about the voter's biometric identification is sent to the

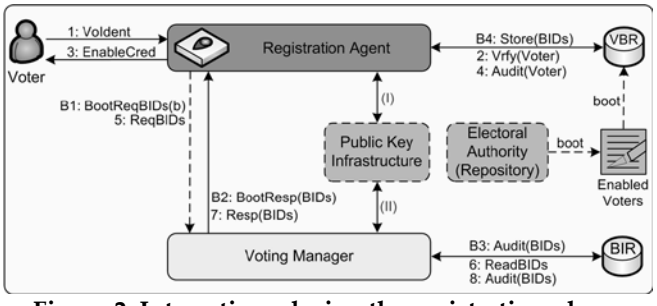


**Figure 2. Interactions during the registration phase**

Voting Manager, ensuring the voter's anonymity. The biometric authentication avoids voter impersonation.

After authenticating the voter, the Voting Console validates the Registration Agent signature in the voter's credential, through the Public Key Infrastructure. Also, Voting Console deciphers the three Ballot IDs (*BID*s) sent by the Voting Manager through the Registration Agent. The Voting Console always takes the first *BID* from the credential, names it *RID* (Receipt Ballot ID), and sends it to the Voting Manager (event 1).

The Voting Manager verifies the Voting Console's signature (event II, figure 3) and queries its Ballot ID and Ballots Repository (*BIR*) to check if the Receipt Ballot ID (*RID)* is valid and was not used before (event 2), to prevent a reply attack [12,13,14]. If the *RID* is valid and the voter did not vote yet, the Voting Manager retrieves the ballot with eligible candidates signed by the Electoral Authority from *BIR*. Then the Voting Manager replicates the ballot, to build a set with three equal ballots.

The Voting Manager logs the *RID* supplied in event 1, to track the voter's activity during the voting phase. However, the Voting Manager does not know the voter identity, which is only known by the Registration Agent, during the registration phase; after that, the *RID* number is the sole identity of an authentic voter in the system.

For each candidate, the Voting Manager puts an initial mark in one randomly chosen ballot in the three ballots set (Ballot 3 for Candidate A, Ballot 2 for Candidate B, Ballot 1 for Candidate C, and Ballot 1 for Candidate D, for instance). After that, the Voting Manager sends the marked ballots to the Voting Console (event 3).

[9], each candidate marked only once in the set of three ballots are not voted; the vote assignment is indicated by two marks in the three ballots set.

The Voting Console can also provide resources to ease the voting procedure, like candidate photographs, search for candidates, by parties, name, number, etc. It can provide a touch-screen interface, a Braille code, speech synthesis of the screen contents, and so on.

After the voter ends voting, the Voting Console provides facilities to ease the vote verification (as a vote summary), and asks the voter to choose a ballot to keep as voting receipt. The Voting Console assigns the chosen ballot with the Receipt Ballot ID (*RID)* and assigns the two other ballots the two remaining *BID*s received from the Registration Agent with the voter's credential. Then, the Voting Console makes a backup copy of the three ballots.

Each one of the three ballots, in random order, is encrypted by the Voting Console using a distinct public key. The Voting Console uses the public keys from Election Representatives – each one responsible for one Ballot Repository: $BR_1$, $BR_2$, and $BR_3$. Then the Voting Console sends the encrypted ballots to the Voting Manager (event 4), which receives them, signs them, and sends each one to a distinct repository (event 5). Each Electronic Ballot Box, when receiving a ciphered ballot, validates the signature of the Voting Manager (event III), stores it at random (e.g. applying a hash function on it), and replies with an acknowledge message, if the storage succeed (event 6). In order to indicate that the three ballots finished the voting phase, the Voting Manager updates its

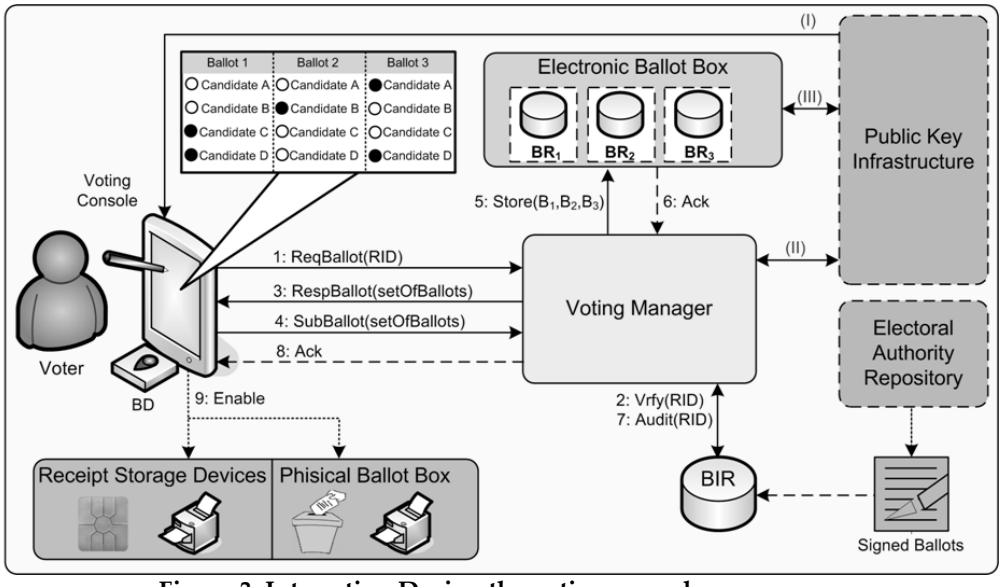


**Figure 3. Interaction During the voting procedure**

This initial ballot marking performed by the Voting Manager eases the voting procedure in the Voting Console. As the Voting Manager already marked randomly all candidates once each in the three ballots set, the voter only needs to put an additional random mark (in an unmarked ballot) for each candidate she intends to vote for (e.g. Ballot 3 for Candidate D in figure 3). According to

Ballot ID and Ballots repository (BIR), marking the corresponding Receipt Ballot ID (*RID)* as used (event 7). The storage of ballots at random in three distinct repositories avoids keeping relationships among the three ballots, assuring the secrecy of the vote.

The Voting Manager informs the Voting Console that the votes are stored in the electronic ballot boxes (event

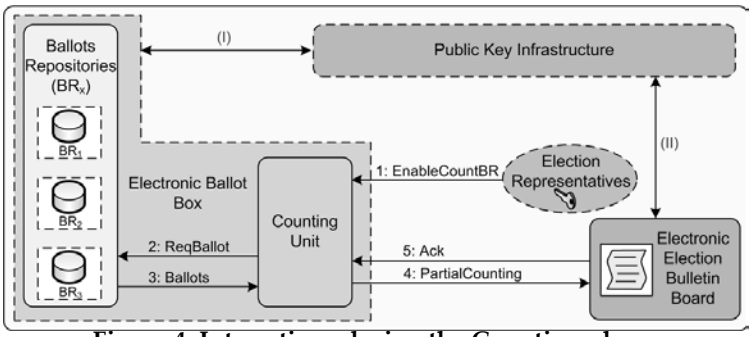


**Figure 4. Interactions during the Counting phase**

8). The Voting Console then takes its backup copy of the vote (3 ballots), encrypts the two ballots that are not bound to *RID* (named here the *unbound ballots*) using the Electoral Authority's public key, and stores them in a persistent storage provided by the voter (a Smart Card or a printer, for example). A clear-text copy of the *RID* ballot is also stored in that Receipt Storage Device (event 9), to serve as a voting receipt. Alternatively, a summarized ballot with no IDs and in a printer-friendly layout, containing only the voted candidate, could be printed and stored in a Physical Ballot Box attached to the Voting Console (event 9). That printed ballot can be used for manual recounting, in the case of election contestation.

Using a physical ballot box could bring problems, because printers can fail and the voter could spend time doing the printed vote verification. Our recommendation for vote materialization is to use a persistent storage as suggested. To provide a vote backup, the unbound encrypted ballots can be encrypted using the Electoral Authority public key, re-encrypted using the voter's public key, and sent to a repository along with the receipt ballot in clear text.

together, they can decrypt her vote using their respective private keys, print a summary of the vote and to put it in a physical ballot box. Such approach could overcome printing problems and verification delays that could arise during the voting day, but preserving the vote secrecy.

### The Vote Storage and Counting Phase

The Electronic Ballot Box is the entity responsible for storing the ballots sent by the Voting Manager, and for computing the vote counting. The Electronic Ballot Box is composed by three Ballot Repositories and a Counting Unit. The Counting Unit manages the votes counting, sending the results to an Electronic Election Bulletin Board, for publication. Each Ballot Repository ($BR_1$, $BR_2$, $BR_3$) is under the responsibility of an Election Representative. This phase is depicted in Figure 4.

As ballots were encrypted using Election Representative's public keys, the Counting Unit only starts counting votes on a Ballot Repository when the corresponding Election Representative provides it her private key. This should happen only after the election finishes, under the coordination of the Election Authority. Election Repre-

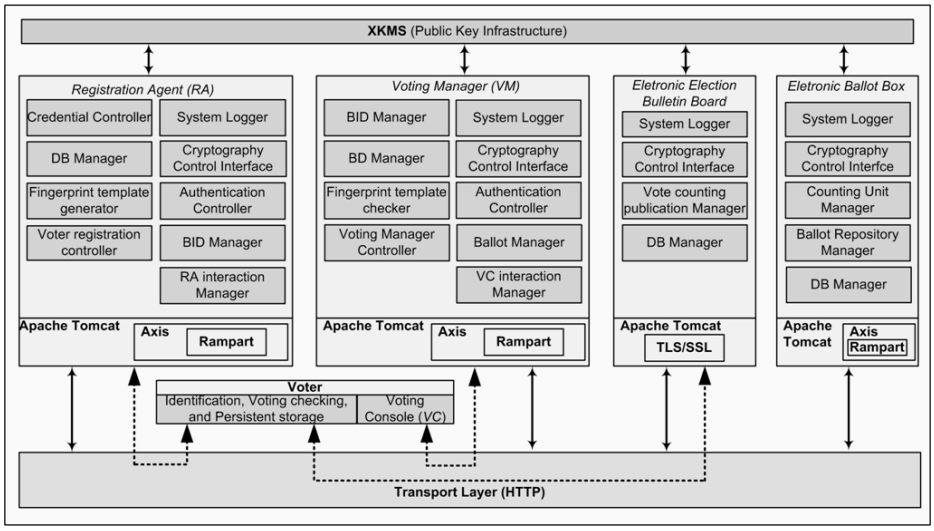


**Figure 5. Prototype Architecture**

The goal of this double encryption is to guarantee that the vote remains inviolable, protected by the voter's public key, and that the voter cannot trade her vote, thanks to the encryption in Electoral Authority's public key. If needed, the voter can meet the Electoral Authority and, sentatives' private keys are valid only for the current election, and are informed to the counting unit on a secured physical media, like a Smart Card (event 1, figure 4). This scheme is adopted to avoid partial counting.

After the counting phase is enabled, the Counting Unit

sends messages requiring all ballots stored in the three repositories (event 2), which are replied by each repository with ballots (event 3).

The Electronic Election Bulletin Board is responsible for receiving vote totals for each candidate from the Electronic Ballot Box. Once the counting starts, partial bulletins are automatically sent to the election bulletin board database and summary reports can be published in a web page.

As an example, partial counting can be computed at several levels, like polling stations, districts, cities, states, and so on. In order to confirm that the election bulletin board received and stored correctly the election bulletins and the receipt list, it replies the Counting Unit with an acknowledge message (event 5).

The list of Receipt Ballot IDs (*RID*s) provided by the Voting Manager gives the information that the Counting Unit needs to identify the votes that should be published on the election bulletin board. Those votes will be checked by voters against their receipts, to ensure that their votes were correctly counted.

Interactions with the Public Key Infrastructure (events I and II, figure 4) include digital signature verification, since all the transactions between entities are signed.

## IMPLEMENTATION

A proof-of-concept prototype was developed using Web Services (WS, http://www.w3.org/TR/ws-arch) and the Election Markup Language (EML, http://www.oasis-open.org/committees/election). Web Services provide standard services and security, while EML provides standard XML schemes to define voting data structures.

EML schemes are organized according three phases: pre-election, election, and post-election. Our prototype uses several EML schemes for each phase. In pre-election, it uses EML schemes 210, 220, and 230 for eligible candidates, and 310 and 330 for enabled voters. The Pre-election phase was not detailed in our proposed architecture.

The prototype modules were developed using Apache Tomcat to run Java Servlets and Java Server Pages http://tomcat.apache.org), and Apache Axis to provide SOAP (Simple Object Access Protocol) support, for communication among system entities. The Apache Rampart module (http://ws.apache.org/axis2) for Axis provides support to WS security (http://www.oasis-open.org/committees/wss). Figure 5 shows the main modules of the prototype, developed as Apache TomCat applications.

During the voting phase, the logging system records all relevant actions of each entity, using the TomCat logging facility. However, relevant information in the voting system, involving the registration, voting, and counting phases (according to the EML 480 scheme) is stored in a database (DB). We adopted an Oracle DB (http://www.oracle.com/database/ berkeley-db/xml) for each repository and XML XPath/XQuery for DB operations.

The cryptography control interface uses Apache Rampart to send XML encrypted and signed messages to verify signatures using the XKMS WS-based PKI (http://www.w3.org/TR/xkms). The Open XKMS (http://sourceforge.net/projects/xkms) initiative was used as PKI implementation in the prototype.

The trust relationship among entities is based in a locally maintained list of trusted public keys, since the Apache Rahas (WS-Trust) and STS (Secure Token Service) facilities are not yet available.

In figure 5, the authentication controller of Registration Agent, the fingerprint template generator, the authentication controller of Voting Manager, and the fingerprint template checker communicate directly with the voter, getting and verifying her identification, which is stored according the EML 420 and 430 schemes.

The Ballot ID (*BID*) manager, along with the Credential controller, provides voting credentials; the voter registration manager implements the core of the Registration Agent. Using the scheme defined in EML 410, the *BID* manager generates the *BID*s and the Ballot Manager makes the initial candidate marks in each three ballot set; the Voting Manager Controller and the Voting Console interaction Manager constitute the core of the Voting Manager.

The Voting Console implementation is a web page running a JSP voting application for the voter.

During the post-election (counting phase), the Ballot Repository manager and the Counting Unit (the core of Electronic Ballot Box), provide the vote counting bulletins that are sent for publication in the Electronic Election Bulletin board. The counting format is defined by EML 510, while publication formats are defined by EML 520. Electronic Election Bulletin Board public access is done through secure (HTTPS) web pages.

If voter coercion and vote trading are real risks, we suggest the Voting Console to be placed in a kiosk under external vigilance, during the election process. Like in conventional elections, the voter should use the voting console alone.

### Design Diversity

We believe that the implementation of an electronic voting system must be done using standard interfaces and design diversity. Indeed, a well known entity must define the requirements and the interfaces for the electronic voting system based on well-known standards. The uses of standards enable developers to design and to implement software components compliant to a system specification.

A homologation process can determine which software is compatible with the adopted standards. Thus, one can select some of the approved software to dynamically build the electronic voting system, without dependency from a single vendor or specific technology.

For instance, in the Election Day, each system module

could be deployed from one component chosen at random from a set of previously homologated components. The same strategy can be applied to all system components, providing better resistance against software fault and tampering.

Several efforts have been done to define computer election standards. For example, the IEEE P-1583 Voting Equipment Standard focuses the development of voting machines, like the Direct Recording Electronic (DRE); the IEEE P-1622 Voting Systems Electronic Data Interchange defines formats and protocols for election data exchange.

## CONCLUSION

This article presents an Electronic Voting System architecture that provides the most relevant requirements and properties expected from such systems in an integrated environment. We found no similar architectures in the technical literature.

The proposed architecture uses asymmetrical key cryptography and responsibility scattering among distinct and distributed entities, in order to guarantee the voting security requirements without creating critical security points.

The paper-based three ballot scheme proposed by [9] to provide vote receipts was adopted in our architecture. The ballot pre-filling simplified the voting procedure, improving its usability, while maintaining its security properties.

The encrypted ballots and the voter receipt constitute a viable alternative to the conventional vote printing, to provide vote materialization. The proposed architecture considers also the participation of election representatives, to improve the election transparency and to ensure the respect to democratic principles.

We built a working prototype using standard well-known technologies and standards, like Web Services, WS-Security, the Election Markup Language, PKI, XML, SSL/TLS, and the Apache TomCat/Axis frameworks. Although our architecture was explained in terms of general elections, it could be used as well to other kinds of elections, in corporate, academic, and other contexts.

## REFERENCES


[1] M. Byrne, K. Greene and S. Everett, "Usability of Voting Systems: Baseline Data for Paper, Punch Cards, and Lever Machines", CHI 2007 Proceedings, Politics & Activism, April/May, 2007.

[2] M. Naor, A. Shamir, “Visual Cryptography”, Eurocrypt 94, p. 1–12.

[3] J. Benaloh and D. Tuinstra, "Receipt-Free Secret-Ballot Elections", In Proceedings of 26th ACM Symposium on the Theory of Computing, pp. 544-553, 1994.

[4] D. Chaum, "Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms", Communications of the ACM - CACM ’81, Vol. 24, No. 2, pp. 84-88, 1981.

[5] B. Schneier, “Applied Cryptography”, Publisher John Wiley & Sons, Second Edition, 1996.

[6] OASIS, "The Case for using Election Markup Language (EML)", Organization for the Advancement of Structured Information Standards, January, 2007.

[7] R. Mercuri, “Electronic Vote Tabulation Checks & Balances”, PhD Dissertation, University of Pennsylvania, 2001.

[8] P. Neumann, "Security Criteria for Electronic Voting", In Proceedings 16th National Computer Security Conference Baltimore, Maryland, September 20-23, 1993.

[9] R. Rivest, W. Smith, "Three Voting Protocols: ThreeBallot, VAV, and Twin", In USENIX/ACCURATE Electronic Voting Technology Workshop, 16th USENIX Security Symposium, Boston, August, 2007.

[10] David Chaum, “Blind Signatures For Untraceable Payments”, Advances in Cryptology, CRYPTO’82, Plenum Publishing, pp. 199-204, 1982.

[11] A. Fujioka, T. Okamoto, K. Ohta, “A practical secret voting scheme for large scale elections”, Advances in Cryptology, AUSCRYPT ’92, in Lecture Notes in Computer Science, vol. 718, 244–251, 1993.

[12] L. Norden, “The Machinery of Democracy: Voting System Security, Accessibility, Usability, and Cost”. Brennan Report, the Brennan Center For Justice at NYU School of Law, 2006.

[13] D. Jefferson, A. Rubin, B. Simons, D. Wagner, "Analyzing Internet Voting Security", Communications of the ACM, Vol. 47, No. 10, 2004.

[14] R. Saltman, "Accuracy, Integrity and Security in Computerized Vote-Tallying", Communications of the ACM, Vol. 3, No. 10, 1988.