A Three-Ballot-Based Secure Electronic Voting System

This article presents a secure electronic voting system integrated in a single architecture—one that addresses vote receipts, uniqueness and materialization of the vote, and voter privacy and anonymity. Our prototype, built using Web services and Election Markup Language, shows the proposal's viability.

REGIVALDO G. COSTA
Brazilian Parliament

ALTAIR O. SANTIN, AND CARLOS A. MAZIERO
Pontifical Catholic University of Paraná

Today, there's a wide understanding that traditional voting systems should be computerized to reduce the vote counting time, provide evidence that a vote is being correctly accounted, reduce fraud, remove errors in filling out ballots, and improve system usability for people with special needs.[1] In fact, E-voting are increasingly replacing traditional paper-based systems. This raises several security issues, given that democratic principles depend on the electoral process's integrity. Providing security to voting systems isn't trivial. Beyond the classic security properties (integrity, confidentiality, and availability), other properties need to be ensured. Some e-voting system requirements seem contradictory, like ensuring voter authenticity and vote anonymity, providing a vote-counting proof while preventing vote trade, allowing voting via the Internet but avoiding voter coercion, guaranteeing the uniqueness of the vote in decentralized voting, allowing vote automation while providing vote materialization, and ensuring auditability in a software or hardware environment that could malfunction.

Existing systems use complex mechanisms to ensure e-voting security requirements, such as using visual cryptography to provide voting receipts,[2] a shared key to decrypt a vote using homomorphic encryption,[3] and mix networks to create anonymous channels to ensure anonymity for the voter and the vote.[4]

Alternatively, we present a proposal 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. Our proposal goes beyond the classic security properties by considering voting receipts, voter coercion, vote trade, vote materialization, voting process auditability, and voter anonymity and authenticity. Our architecture also considers the participation of election representatives to improve election transparency and ensure the respect of democratic principles. We built a proof-of-concept prototype using Web services and the Election Markup Language (EML, a proposed standard for election data[6]) to show the proposal's viability.

Voting system requirements

Each country defines a set of specific laws to guide its voting system, to establish its organization, and to ensure its impartiality, integrity, and democratic principles. Elections based in an e-voting system must comply with the laws and rules for voting systems, and also fulfill the following requirements (as discussed elsewhere[5,7,8]):

- *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.* Final vote counting must exactly represent the number of voters (vote uniqueness) and their intent (quality of the vote).
- *Availability.* An e-voting system should guarantee voting service availability and respect to its security requirements during the entire election process.
- *Authenticity.* Voter authenticity must be verified at two distinct phases: at voter registration and just be-

fore the voting procedure; the system should provide means to prevent voter impersonation.
- *Anonymity.* The vote must remain anonymous during the voting process; afterward, there should be no way to associate a vote to its voter, or vice-versa.
- *Vote receipts.* In an e-voting system, vote counting is done computationally—not under the direct observation of poll watchers. Therefore, vote receipts are needed, to allow voters check if their votes were correctly counted while preventing practices like voter coercion and vote trading.
- *No vote trading.* No voter should have access to material evidence that certifies to other people the quality of his or her vote.
- *No voter coercion.* No party should have means to impose on voters to vote against their intent.
- *Uniqueness.* Voters should be able to vote only once in the same election.
- *Auditability.* A voting system must provide an audit trail of the entire voting process for detecting fraud, software or hardware malfunction, or human operation errors. However, such information can't keep information that compromises other security requirements.
- *Usability.* An e-voting system should be user-friendly, offering visual, touch, and audio resources that allow the voter to vote quickly without help from others.

Our proposal fulfills such properties, as we'll see in the following sections. It also provides vote materialization—the system is able to materially reproduce the quality of each vote, allowing manual vote recounting, if requested.

The architecture

Given that today's E-voting aren't yet mature, researchers are still proposing new paper-based systems. Such systems introduce properties not present in conventional ones, such as vote receipts.

Our fully computerized architecture adopts the three-ballot scheme from a paper-based voting system proposed in other work.[9] That scheme uses three equal ballots for each vote, with each one having a unique numeric identifier. The voter checks off his or her candidates in two ballots; for all the other candidates, just one check is needed, on one of the three ballots, randomly. This way, the candidates 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 as a vote receipt. The three ballots are then stored. After the election, the electoral authority publishes all copied ballots to let voters verify whether their votes were accounted for.

Figure 1 presents an overview of our proposed architecture. We built it using the following entities: a registration agent, a voting console, a voting manager, an electronic ballot box, and an electronic election bulletin board.

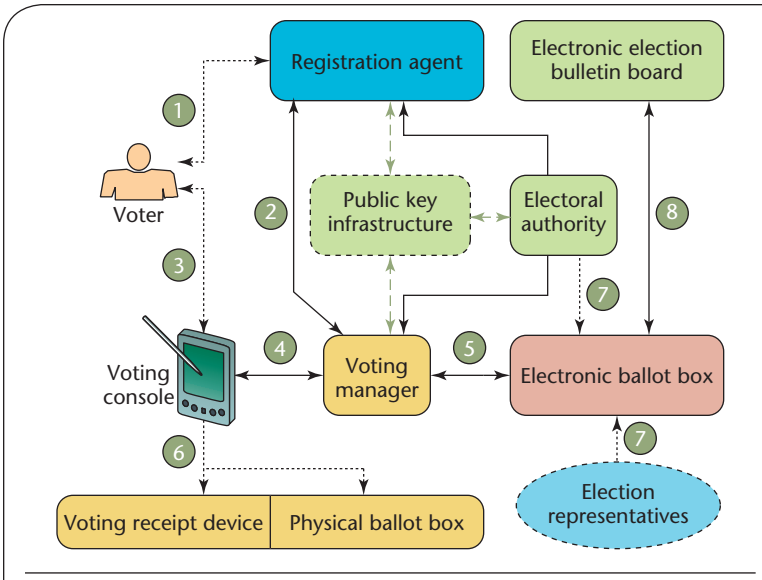

Figure 1. Overview of the proposed architecture. It uses classic cryptography techniques and a standard public key cryptosystem to ensure its security properties. Also, the architecture entities are scattered to separate their responsibilities, avoiding critical security points.

To vote, voters present themselves to the registration agent to get a credential that qualifies them to vote (event 1 in Figure 1). The registration agent interacts with the voting manager to obtain the corresponding ballot IDs (BIDs; event 2) and uses them to build credentials that are returned to the voters. Later, after authentication (event 3), voters use the voting console to vote (event 4), and the voting manager stores the votes in the electronic ballot box (event 5) while the voting console gives a voting receipt back to each voter (event 6). When the election finishes, the electoral authority and election representatives start the counting phase (event 7), counting the votes and publishing the receipts in the electronic election bulletin board (event 8).

Our architecture considers the voters, the election representatives, and an electoral authority as actors. In a general election, election representatives can be persons from the civil society and political parties, who are responsible for watching the voting process. The electoral authority manages the electoral process and enforces the voting rules and laws.

The voting process consists of three phases: voter registration, the voting itself, and vote storage and counting.

The registration phase

The registration agent is responsible for voters' admission and qualification during the registration phase,

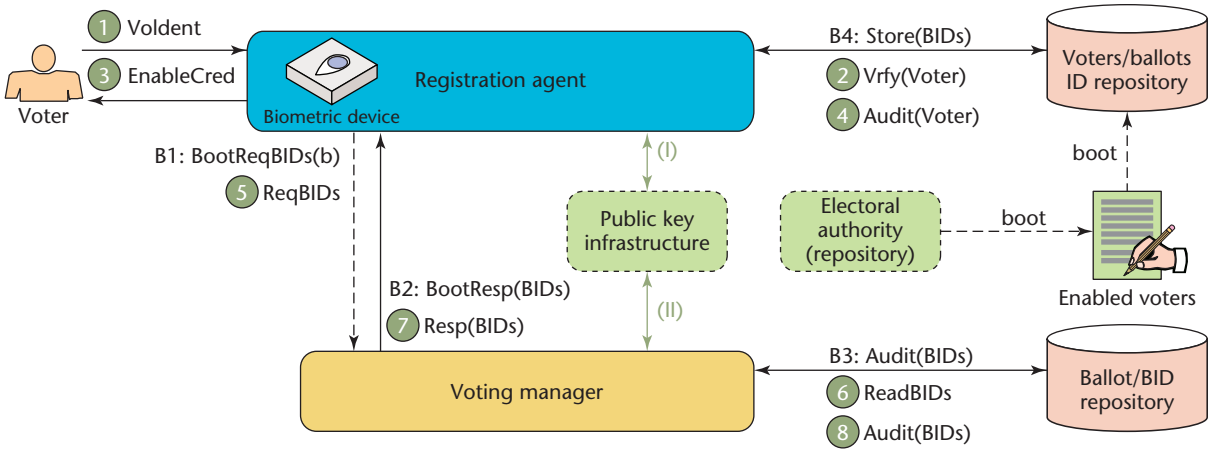

Figure 2. Overview of the registration phase. During this phase, a registration agent manages the admission and qualification of voters in the system and applies a blind signature scheme on the credentials provided to voters. Such a scheme ensures voter anonymity, preventing the binding between voters and their credentials, even if the registration agent is compromised.

depicted in Figure 2. Its tasks include receiving voters at the polling station and requesting their identification (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 the agent's initialization (boot), it starts a voters/ballots ID repository (VBR) using data from a repository of voters that the electoral authority maintains. The registration agent also requests b BIDs from the voting manager (events B1 and B2 in Figure 2) and stores them locally in the VBR (event B4); b can be defined by each electoral authority. The voting manager logs the supplied BIDs to the registration agent in a local repository for BIDs and ballots (BIR; event B3). This initialization procedure makes the sequence of voters accessing the registration agent unpredictable (for the voting manager) so as to prevent voter anonymity violations.

Once the registration phase starts, voters should identify themselves to the registration agent (event 1, Figure 2), who then verifies whether the voter can vote (event 2), querying the 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 to the voter (event 3). At the same time, it updates the repository of voters (event 4) to register that the voter was qualified to vote to assure vote uniqueness.

To keep b BIDs in its voters/ballots ID repository, the registration agent requires three new BIDs from the voting manager (event 5), which chooses three new BIDs (event 6), and ciphers each one separately using the voting console's public key. Next, the voting manager sends them back to the registration agent (event 7), and logs the BIDs in the local BIR repository (event 8).

If the voting system uses biometric authentication (event 1 in Figure 2), a template of the voter's fingerprint is obtained by the Biometric Device (BD), ciphered using the voting console's public key, and attached to the credential (event 3). Such a scheme guarantees the voter's authenticity and prevents fraud related to impersonation during the voting phase.

The random BIDs that the registration agent sends in event 3 (Figure 2) consist of three IDs that the voter will use during the voting process. The registration agent doesn't know them because, as mentioned, 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 voter's credential.

Interactions with the public key infrastructure (events I and II) include procedures for signature authenticity verification given that all the transactions between entities in the entire process are digitally signed.

The voting phase

The voting console interacts with the voter during the voting phase (Figure 3). Therefore, all messages from the voting console to the voting manager result from interactions between the voter and the voting console.

If the voting system adopted biometric authentication during voter registration, the voting console receives the voter's biometric template, decrypts it, and verifies its authenticity (event I, Figure 3) through the registration agent's digital signature. Then, the voting console requests a voter's fingerprint using a biometric device. Verification consists of comparing the template obtained from the device with the template coming from the registration agent. No information about the voter's biometric identification is sent to the voting manager, ensuring

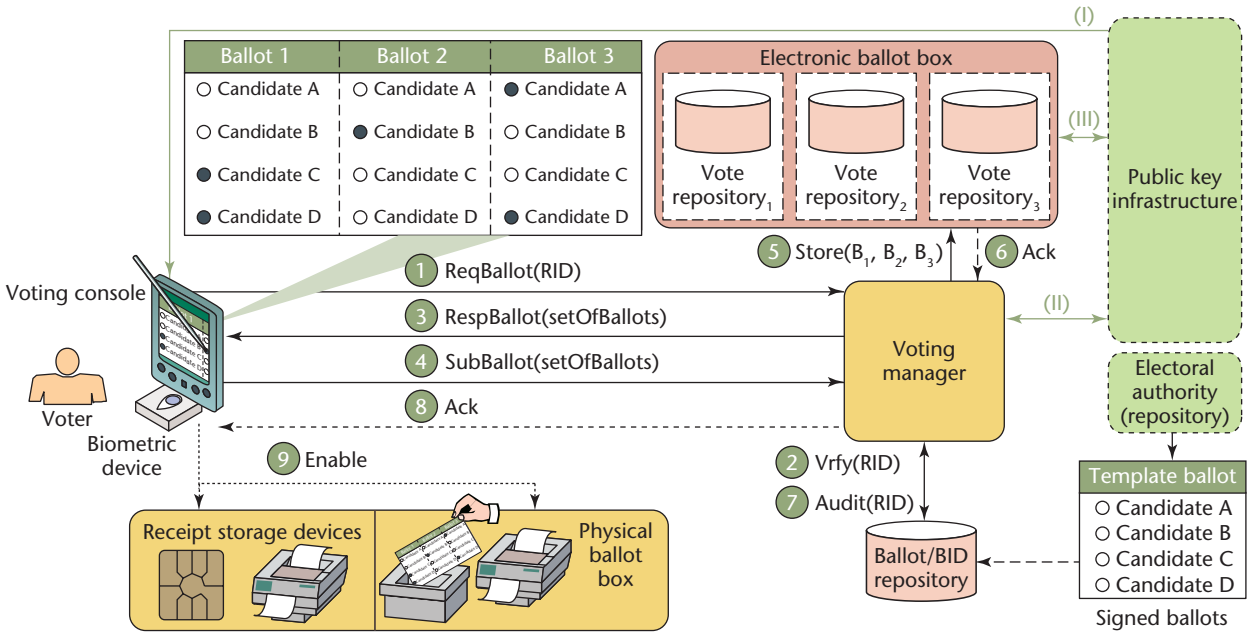

Figure 3. The voting phase. The central entities during the voting phase are the voting manager and the voting console. The voting manager is responsible for the coordination aspects of the voting phase, authenticating the voters, providing the ballots for filling out, and storing the votes. The voting console builds the voter interface, verifies the voter qualification, makes possible the vote materialization, and generates the vote receipts.

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. It also deciphers the three BIDs sent by the voting manager through the registration agent. The voting console always takes the first BID from the credential, names it as 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 BIR to check if the RID is valid and wasn't used before (event 2) to prevent a reply attack.[12,13] If the RID is valid and the voter hasn't yet voted, the voting manager retrieves the ballot with eligible candidates signed by the electoral authority from BIR. The voting manager then replicates the ballot to build a set with three equal ballots.

The manager logs the RID supplied in event 1 to track the voter's activity during the voting phase. However, it doesn't know any voter's identity—that's known only 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).

This initial ballot marking that the voting manager performs simplifies the voting procedure in the voting console, thus improving the usability of the three-ballot scheme. Because the voting manager already randomly marked all candidates once each in the three ballots set, voters need only to put an additional random mark (in an unmarked ballot) for each candidate they intend to vote for (for example, Ballot 3 for candidate D in Figure 3). As noted earlier, each candidate that's marked only once in the set of three ballots isn't voted on; the vote assignment is indicated by two marks in the three ballots set. [9] The voting console can also provide resources to ease the voting procedure, such as candidate photographs, candidate search, a touch-screen interface, Braille code, and speech synthesis of the screen contents.

After the voter votes, the voting console provides facilities to ease vote verification (as a vote summary) and asks the voter to choose a ballot to keep as a voting receipt. The voting console assigns the chosen ballot with the RID and assigns the two other ballots the two remaining BIDs received from the registration agent with the voter's credential. Then the voting console makes a backup copy of the three ballots.

The voting console encrypts each of the three ballots, in random order, using a distinct public key, the ones from election representatives—each one re-

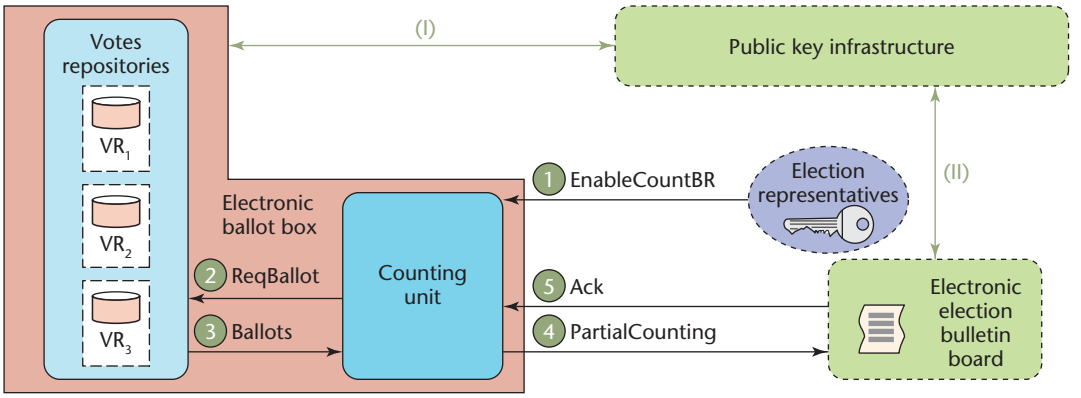

Figure 4. Overview of the vote counting phase. Here, the votes are counted and published in a bulletin board. The election representatives should provide their credential to allow the counting to proceed once the election phase finishes. Altogether with the vote tabulations, vote receipt IDs are also published, allowing each voter to check if his or her vote was correctly counted.

sponsible for one vote repository: VR1, VR2, and VR3. The voting console then 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 voting manager's signature (event III), stores it at random (by applying a hash function to it), and replies with an acknowledge message if the storage succeeded (event 6). To indicate that the three ballots finished the voting phase, the voting manager updates its BIR, marking the corresponding RID as used (event 7). Storing ballots at random in three distinct repositories avoids keeping relationships among the three ballots, assuring the vote's secrecy.

The voting manager informs the voting console that the votes are stored in the electronic ballot boxes (event 8). The voting console then takes its backup copy of the vote (three ballots), encrypts the two ballots that aren't bound to RID (named here the *unbound ballots*) using the electoral authority's public key, and puts them in storage provided by the voter (such as a smart card) or a printer. A clear-text copy of the RID ballot is also stored in the receipt storage device (event 9) to serve as a voting receipt. Alternatively, a summarized ballot with no IDs in a printer-friendly layout that contains 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 if the election results are contested.

Using a physical ballot box could bring problems because printers can fail and the voter could spend time verifying the printed vote. For vote materialization, we recommend using a persistent storage. To provide a vote backup, 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.

The goal of this double encryption is to guarantee that votes remain inviolable, protected by voters' public keys, and that voters can't trade their votes, thanks to encryption in the electoral authority's public key. If needed, a voter can meet the electoral authority and together they can decrypt the vote using their respective private keys, print a summary of it, and put it in a physical ballot box. This approach could overcome printing problems and verification delays arising during voting but preserve vote secrecy.

The vote storage and counting phase

The electronic ballot box is responsible for storing the ballots sent by the voting manager and for computing the vote counting. The electronic ballot box consists of three vote repositories (VR) and a counting unit. The counting unit manages the vote counting and sends the results to an electronic election bulletin board for publication. Each vote repository (*VR*1, *VR*2, *VR*3) is under the responsibility of an election representative. Figure 4 depicts this phase.

As ballots are encrypted using the election representative's public keys, the counting unit starts counting votes on a VR when the corresponding election representative provides her private key. This happens only after the election finishes under the coordination of the election authority. Election representatives' private keys are valid only for the current election and are informed to the counting unit on 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). Each repository containing ballots replies to the messages (event 3).

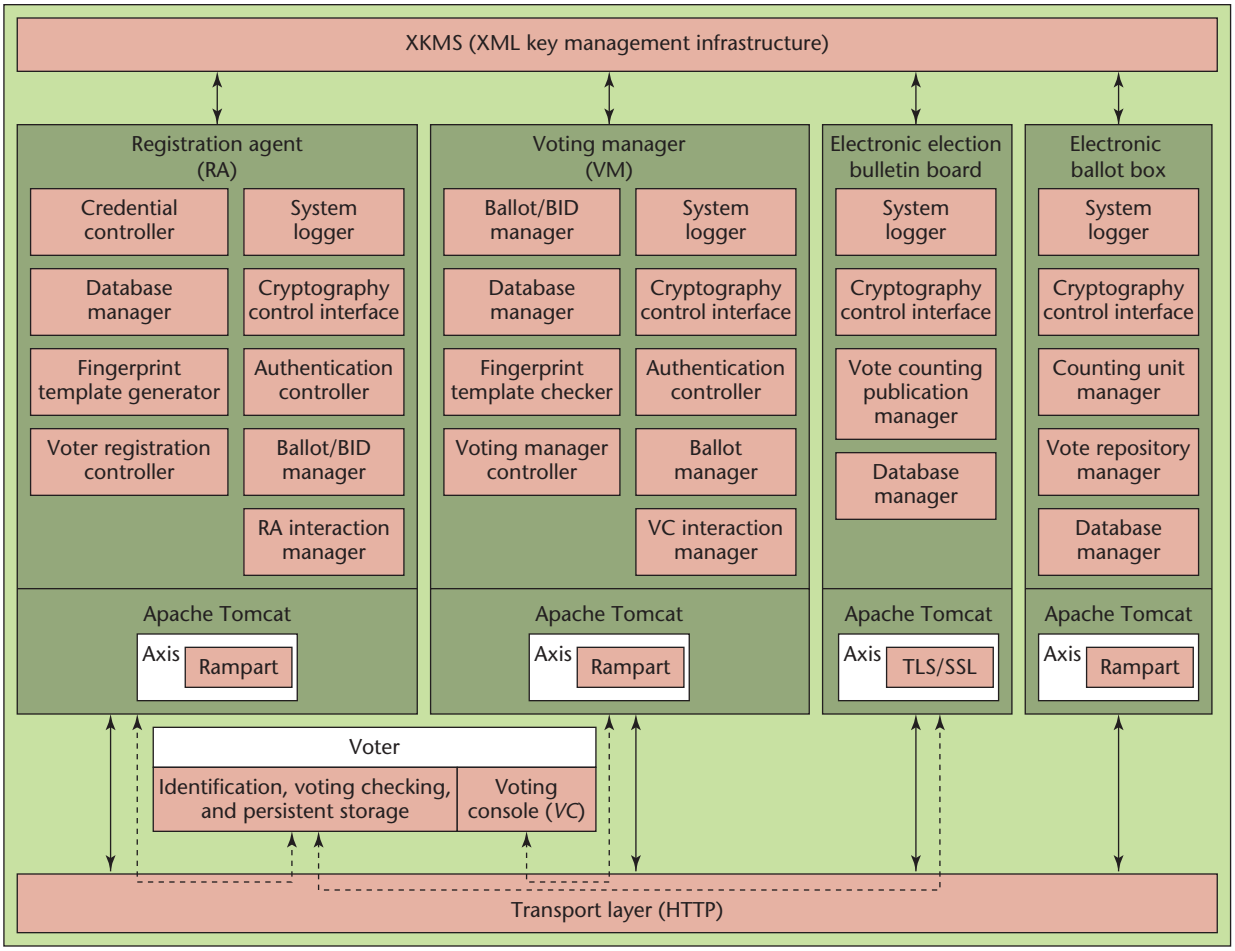

Figure 5. Prototype architecture. The prototype is composed by four Web Services modules and an embeddable XML database. Each Web service implements a distinct entity of the voting system. The user interface is offered through a Web page.

The electronic election bulletin board receives vote totals for each candidate from the electronic ballot box. Once counting starts, partial bulletins are automatically sent to the election bulletin board database. Summary reports can be published in a Web page. For instance, partial bulletins can be computed at several levels, like polling stations, districts, cities, and states. To confirm that the election bulletin board received and stored the election bulletins and vote receipt list correctly, it replies to the counting unit with an acknowledge message (event 5).

The list of RIDs provided by the voting manager gives the information the counting unit needs to identify votes that should be published on the election bulletin board. Voters will check those votes 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, given that all the transactions between entities are signed.

Implementation

We developed a proof-of-concept prototype using Web services (www.w3.org/TR/ws-arch) and EML (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 to 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. We didn't detail the pre-election phase in our proposed architecture.

We developed the prototype modules using Apache Tomcat to run Java Servlets and Java Server Pages (tomcat.apache.org) and Apache Axis to provide SOAP (Simple Object Access Protocol) support, for communication among system entities. The Apache Rampart module (ws.apache.org/axis2) for Axis provides support to WS security (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 registration, voting, and counting phases (according to the EML 480 scheme) is stored in a database. We adopted an Oracle database (www.oracle.com/database/berkeley-db/xml) for each repository and XML XPath/XQuery for database operations.

The cryptography control interface uses Apache Rampart to send XML-encrypted and -signed messages to verify signatures using the XML Key Management System (XKMS, www.w3.org/TR/xkms). The Open XKMS (sourceforge.net/projects/xkms) implementation was used in the prototype.

The trust relationship among entities is based in a locally maintained list of trusted public keys given that the Apache Rahas (WS-Trust) and STS (Secure Token Service) facilities aren't 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 voters, getting and verifying their identification, which is stored according to the EML 420 and 430 schemes.

The 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 BIDs, 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 vote repository manager and the counting unit (the core of the electronic ballot box) provide vote counting bulletins sent for publication in the electronic election bulletin board. EML 510 defines the counting format, whereas EML 520 defines the publication formats. Secure Web pages provide electronic election bulletin board public access.

If voter coercion and vote trading are real risks, we suggest the voting console 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 an e-voting system should be implemented using standard interfaces and design diversity. Indeed, a well-known entity must define the requirements and interfaces for the e-voting system based on well-known standards. The use of standards enables developers to design and implement software components compliant to a system specification.

A homologation process determines which software is compatible with the adopted standards. Thus, one can select approved software to dynamically build the electronic voting system without depending on a single vendor or specific technology.

For instance, on Election Day, each entity 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 tried to define computer election standards. For example, the IEEE P-1583 voting equipment standard focuses on the development of voting machines like the direct recording electronic (DRE) ones. The IEEE P-1622 voting systems electronic data interchange defines formats and protocols for election data exchange.

We believe that our proposal should be deployed in a real environment, to evaluate its behavior in a large-scale experiment. In such an environment, it will be possible to better evaluate its usability, flexibility, and scalability.

As our proposal is structured as distributed components interacting through Web services, it can be used to support elections in a large geographical area with small deployment costs. If vote trading and voter coercion aren't considered a problem, or if reliable mechanisms can be used to avoid them, our proposal can be also used to support Internet-based elections.

Finally, although we explained our architecture in terms of general elections, it could be used as well for 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 Proc., Politics & Activism*, ACM Press, vol. 1, 1997, pp. 171–180.
2. M. Naor and A. Shamir, "Visual Cryptography," *Advances in Cryptology*, Eurocrypt 94, Springer, vol. 950, 1995, pp. 1–12.
3. J. Benaloh and D. Tuinstra, "Receipt-Free Secret-Ballot Elections," *Proc. 26th Ann. ACM Symp. Theory of Computing*, 1994, ACM Press, pp. 544–553.
4. D. Chaum, "Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms," *Comm. ACM*, vol. 24, no. 2, 1981, pp. 84–88.

5. B. Schneier, *Applied Cryptography,* 2nd ed., John Wiley & Sons, 1996, pp. 125–133.
6. Oasis, "The Case for Using Election Markup Language (EML)," white paper, Oasis Election and Voter Services TC, 2007; www.oasis-open.org/committees/election.
7. R. Mercuri, *Electronic Vote Tabulation Checks and Balances*, doctoral dissertation, Dept. of Computer and Information Systems, Univ. of Pennsylvania, 2001.
8. P. Neumann, "Security Criteria for Electronic Voting," *Proc. 16th Nat'l Computer Security Conf.*, US Nat'l Inst. of Standards and Technology; www.csl.sri.com/users/neumann/ncs93.html.
9. R. Rivest and W. Smith, "Three Voting Protocols: ThreeBallot, VAV, and Twin," *Usenix/Accurate Electronic Voting Technology Workshop, 16th Usenix Security Symp.*, 2007; http://people.csail.mit.edu/rivest/RivestSmith-ThreeVotingProtocolsThreeBallotVAVAndTwin.pdf.
10. D. Chaum, "Blind Signatures for Untraceable Payments," *Advances in Cryptology* (Crypto 82), Plenum, 1982, pp. 199–204.
11. A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large-Scale Elections," *Advances in Cryptology* (Auscrypt 92), LNCS, vol. 718, 1993, pp. 244–251.
12. L. Norden, "The Machinery of Democracy: Voting System Security, Accessibility, Usability, and Cost," *Brennan Report*, The Brennan Center for Justice, 2006.
13. D. Jefferson et al., "Analyzing Internet Voting Security," *Comm. ACM*, vol. 47, no. 10, 2004, pp. 59–64.

Altair O. Santin *is an associate professor at the Pontifical Catholic University of Paraná State. His research interests include security in distributed systems, access control, and public key infrastructure. Santin has a PhD in electrical engineering from the Federal University of Santa Catarina, in Brazil. He is a member of the IEEE Computer Society, the ACM, and the Brazilian Computer Society. Contact him at santin@ppgia.pucpr.br.*

Regivaldo G. Costa *is a security analyst of the Brazilian Parliament's Electronic Voting System Division, Chamber of Deputies. His research interests include network security and management, distributed systems, and grid computing. Costa has a Master's Degree student in distributed systems research at the graduate program in computer science of the Pontifical Catholic University of Paraná State. Contact him at regivaldo.costa@camara.gov.br.*

Carlos A. Maziero *is a full professor at the Pontifical Catholic University of Paraná State. His research interests include security and resource management in operating systems. Maziero has a PhD in computer science from IRISA (University of Rennes I) in France. He is a member of the ACM, the Brazilian Computer Society, and is the Brazilian delegate at the IFIP Technical Committee TC11. Contact him at maziero@ppgia.pucpr.br.*