5-2017

XML Based Security Model for Enhancing the Integrity and the Privacy of E-Voting Systems

Imali Anushka Arthanayaka
imali25@yahoo.co.uk

XML Based Security Model for Enhancing the Integrity and the Privacy of E-Voting Systems

by

Imali A. Arthanayaka

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

in Information Assurance

May, 2017

Thesis Committee:
Susantha Herath, Chairperson
Changsoo Sohn
Jayantha Herath

Abstract

As the world is becoming more technological, using electronic voting could be very beneficial in elections rather using traditional paper-based election schemes. However, there are many security related issues that can cause significant problems in electronic voting (e-voting). Violating voters' privacy or integrity of ballots would definitely cause serious problems with the entire election process. People may refuse to accept the electronic form of elections. Existing e-voting systems use sophisticated but inefficient, and expensive techniques to satisfy the security requirements of e-voting. Therefore, most of small and mid-size electoral populations cannot employ e-voting systems in their elections and experience remarkable benefits of e-voting. In this thesis, a new electronic voting approach is proposed using extensible markup language (XML) to verify and secure the integrity as well as to preserve the privacy of the voters. The evaluation results of this thesis show that the new approach is an implementation friendly, efficient, and also cost-effective approach to safeguard integrity and privacy related security requirements of e-voting systems for small and mid-size electoral populations.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my adviser, Prof. Susantha Herath, for his continuous support throughout my master studies and research, his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I would like to thank the rest of my thesis committee: Prof. Jayantha Herath and Prof. Changsoo Sohn for their encouragement and insightful comments.

I acknowledge the people who mean a lot to me, my parents for showing faith in me and giving me liberty to choose what I desired. I salute you all for the selfless love, care, pain, and sacrifice you did to shape my life. I would never be able to pay back the love and affection showered upon by my parents. Also I express my thanks to my brothers and sisters for their support and love.

Finally, I owe thanks to a very special person, my husband, Charitha Hettiarachchi for his continued and unfailing love, support, and understanding during my pursuit of master degree that made the completion of this thesis possible. You were always around at times I thought that it is impossible to continue, you helped me to keep things in perspective. I appreciate my loving daughter, Ranolee for abiding my ignorance and the patience she showed during my thesis writing. Words would never say how grateful I am to both of you.

Table of Contents

List of Tables

List of Figures

Figure Page

Chapter I: Introduction

Introduction

As computer and Internet technologies emerged most traditional paper-based procedures, activities, and systems were replaced by electronic systems. As a result, most of the important systems like banking systems, hospital systems, and airline systems started to go online, and providing services to their customers that were efficient and accurate. Therefore, it is not a surprise that people thought about introducing the same convenient facilities to their traditional voting systems.

As a result, people started inventing electronic voting systems (EVS) in different ways (Farivar, 2012). When using EVS, there are not only important advantages but also many security related issues found in such systems. Confidentiality, integrity, privacy, and availability are some of the major aspects needed to be assured when using EVS (Ibrahim, Kamat, Salleh, & Aziz, 2003). It is very difficult to guarantee these properties in e-voting system, since these types of systems are very much prone to cyber-attacks, such as denial-of-service attacks. Cyber attackers or any other hacker may try to diminish the security of EVS to obtain sensitive ballot data for many reasons, such as financial benefits.

Introducing new safety procedure and approaches to e-voting are therefore important. Those new approaches will help improve not only the security of the system but also the quality of the EVS. From years of use and experience with the traditional paper ballot approach, we already know that it is a safe and secure system to a great extent. Therefore, the electronic form of voting must guarantee that it is at least as safe as current traditional system. In this thesis, combination of cryptographic and XML technologies were introduced to EVS in order to secure integrity and privacy properties of EVS in an affordable and efficient manner.

The rest of the thesis is organized as follows: Chapter II describes the related work. Chapter III describes the design of this thesis in detail. Chapter IV describes the implementation of the prototype system. Chapter V presents the results and evaluation. Finally, Chapter VI presents conclusions and discusses possible future work.

Problem Statement

Existing e-voting systems use complex and expensive approaches to maintain the security attributes such as integrity and privacy. Therefore, it is not affordable to use electronic voting in small and mid-size electoral populations. Moreover, those expensive systems can only be implemented on specific platforms and may not efficiently process electronic ballots.

Nature and Significance of the Problem

Electronic voting has substantial advantages over traditional paper-based voting, such as increase in voter turnout, fast, convenience, and accelerate the decision making process. However, because of the complex methods involved and the requirement of expensive infrastructure to secure the EVS, most small and mid-size electoral populations cannot use electronic voting systems and miss the opportunity to obtain the aforementioned significant advantages. This thesis study was useful in finding new approaches to minimize the abovementioned barriers when using electronic voting in Small and mid-size electoral populations.

Objective of the Study

The objective of this thesis is to introduce a new approach to secure the integrity and privacy of electronic voting systems by using a cost-effective and efficient approach which can be implemented on any platform.

Limitations of the Study

The proposed new approach of this thesis focused only on two security attributes of electronic voting systems, namely, integrity and privacy. In addition, the implementation of the proposed approach highly focused on small and mid-size electoral populations rather large scale elections.

Summary

This chapter discussed the importance of electronic voting and current security related issues of EVS. Moreover, this chapter briefly discussed the significance of the proposed new approach and its limitations. The next chapter discusses in detail the literature related to the security issues of EVS, critical security attributes, and software security technologies.

Chapter II: Background and Review of Literature

Introduction

In this chapter, the background of e-voting systems, their security related issues, current technologies that support the safeguarding and maintenance of the security of software systems are discussed. Furthermore, the chapter discusses in detail the cutting-edge technologies used in the proposed new approach in order to protect the security of EVS.

Background Related to the Problem

Most of countries continue research on advance e-voting systems because it offers an extraordinary set of advantages that cannot simply be ignored. Countries like, Australia and Canada have already used e-voting in some parts of their countries while Estonia has used it nationwide. Figure 1 shows how Internet Voting was used around the world by 2012 (Esteve, Goldsmith, & Turner, 2012). Figure 2 visualizes how the countries around the world either extended or discontinued the implementation of electronic voting by 2015 (Krimmer, 2015). According to the Figure 2, most countries have significantly expanded the use of electronic voting while a few countries discontinued electronic voting, primarily due to software security concerns.

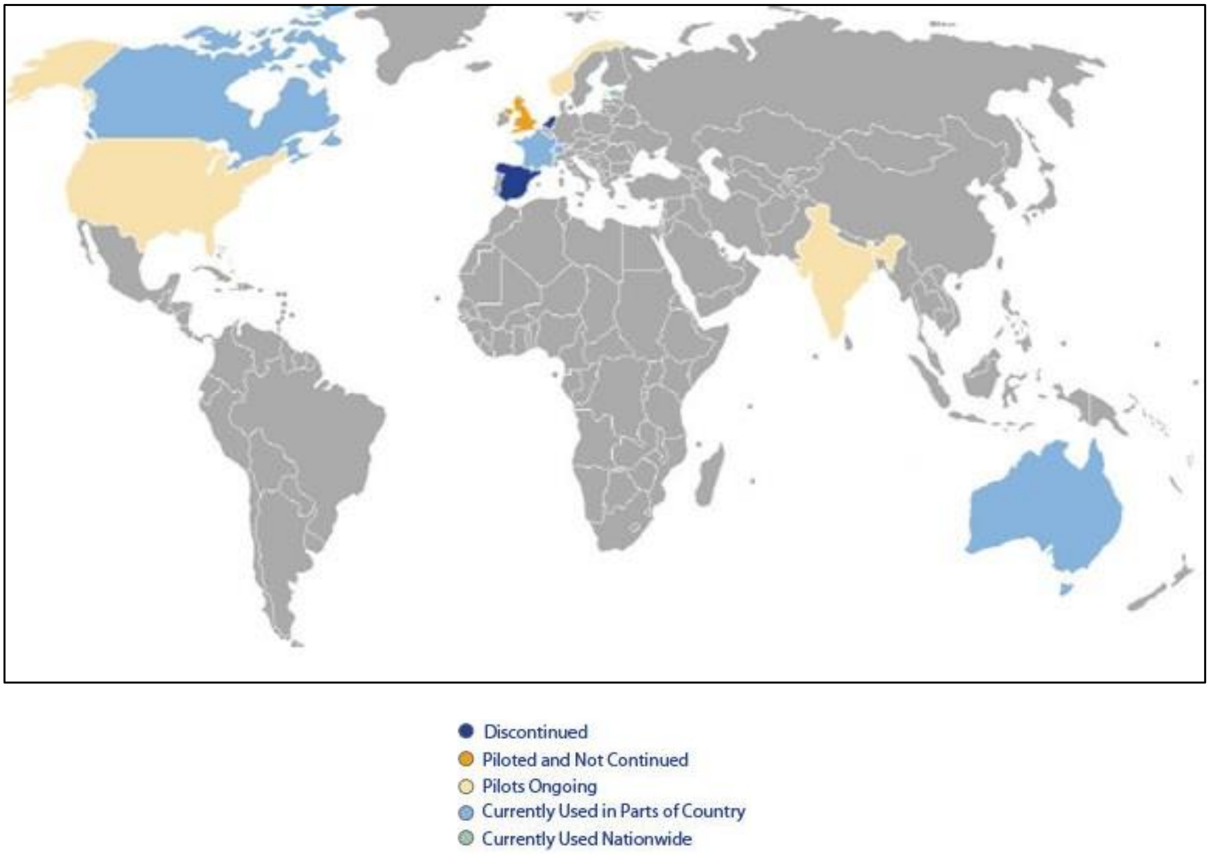
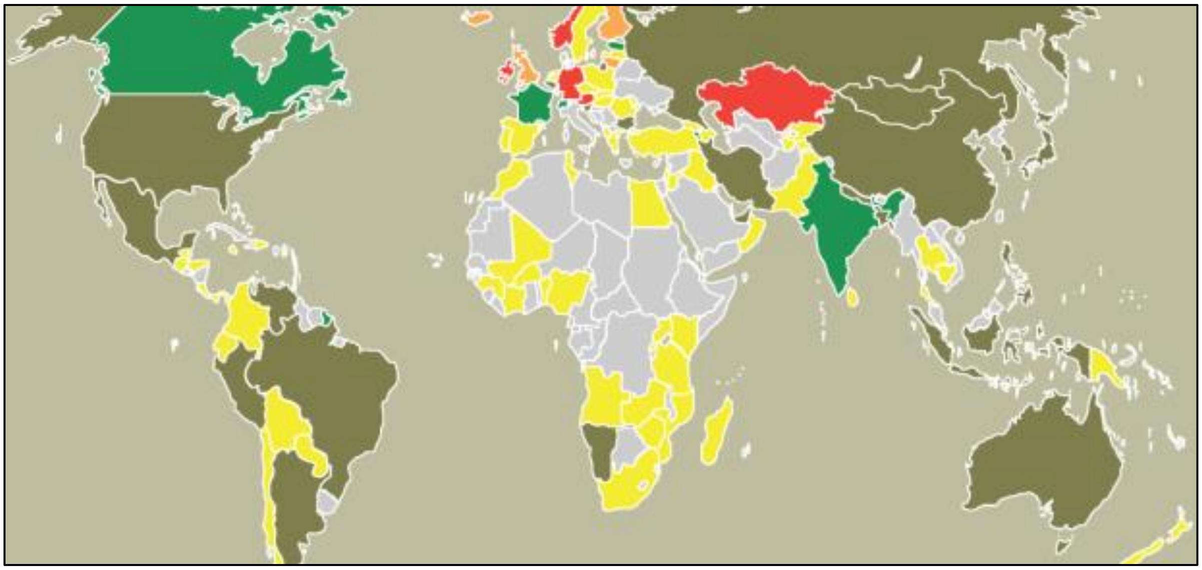


Figure 1. Internet voting around the world by 2012 (Esteve et al., 2012, p. 14).



Grey: no e-voting
Yellow: discussion and/or voting technology pilots
Orange: Discussion, concrete plans for Internet voting
Dark green: Ballot scanners and/or Electronic Voting Machines (legally binding)
Green: Internet voting (legally binding) (also used with other voting technologies)
Red: Stopped use of voting technologies

Figure 2. Electronic voting around the world by 2015 (Krimmer, 2015).

Types of E-Voting Systems

Basically voting systems are divided into two broad categories, namely paper ballot and e-voting. Electronic voting systems are further divided into several other types as follows.

Telephone voting. Telephone voting systems allow people to cast their vote through a telephone connection. Voters can select option according to the instruction they hear and even use the key pad to make selections. Some systems are capable of recognizing voice as well. Voter verification is very difficult with telephone voting systems (Technology and the Voting Process, 2014).

Optical scanner. With this system, paper ballots are read by means of an optical scanner. Candidate options are indicated with numbered circles and a voter is supposed to fill the circle according to their choice. Then the paper ballots are fed into a ballot box (Electronic Voting Systems, 2014).

Internet voting. With the advance of the Internet, voting through the Internet has become very popular. Nowadays, most organizational small to midlevel polls are conducted through the Internet. For instance, universities are using Internet Voting for their student body elections because it provides greater flexibility to cast their vote while they stick to their tight schedules. Most of the issues with other voting systems can be avoided with Internet voting, such as voter verification difficulties, privacy and integrity related issues (Electronic Voting Systems, 2014).

Direct recording electronic voting systems (DRE). This is another popular way of voting by means of computers. Most often interfaces of these machines are equipped with buttons or a touch screen. DREs are capable of issuing results quickly. There is a low risk of machine failures. With DREs, e-voting systems developers can provide facilities to voters to customize the interfaces for their convenience. When the ballots (vote data) are transmitted from a polling booth to another location through public network, then it is called “Public network DRE voting system” (Wolf, Nackerdien, & Tuccinardi, 2011).

Advantages of E-Voting Systems

Electronic voting systems can simply be altered to be used with different type of elections, with minimal modification cost. For example, if a university builds an e-voting system for its student body general election, the same system can be modified to be used in a university presidential election. Moreover, e-voting systems support the election process to maintain the integrity of ballots by imposing rules and restricting invalid entries to the system through the electronic ballot. For instance, if the election expects the voters to type only three candidates' names, system can restrict it to definitely three candidates' names. Though voters may accidentally want to put additional names, they won't be able to do so.

With e-voting, it will be much easier to access the voting system. The disabled or handicapped person can be able to vote even without leaving their home. When voting is as easy and as accessible as this, participation rates would increase much more (Gerlach & Gasser, 2009). In addition, e-voting systems can be designed in a way that is very user friendly and very informative to its users (i.e., election administrators, voters, etc.). For instance, if a particular entry is confusing, the system can display "More Information" icons or even pop up messages that can guide its users to eliminate any mistakes. These pop up error messages typically convey what exactly went wrong and may suggest how to avoid potential problems. Depend on the conflicts situations, it can show customized messages which is not possible with a paper format ballot. It would vastly reduce the amount of paper being used, compared to using the traditional paper ballot.

Live validation is another great benefit comes with e-voting system. The system is capable of validating user input as the user progresses through the system. During the user casting his/her vote, the system can validate its input behind the screen and let the user know about any invalid inputs and make corrections before the voter casts the vote and leaves the polling booth. In this way the user has an opportunity to assure his/her vote's validity and acceptance for the current election before even he or she logs out of the system (Voter Validation Process FAQs, 2017).

Disadvantages of E-Voting Systems

Cost of the system. The election authority may have to spend millions of dollars to introduce new e-voting systems. Some countries might need a certification from higher authority to use this kind of systems in public elections. If that happens rigorous amount of testing is required to get the certification and it also may significantly increase the expenses (Wolf et al., 2011).

Security threats. Electronic voting systems are vulnerable to several different types of attacks. A minor security hole would be enough to compromise the whole system and deliberately change casted votes or even delete all voting data. Denial-of-services attacks can be launched to disrupt the availability of the system while Trojan applications can be used to breach confidentiality of ballot data and the privacy of voters. Even operating systems or e-voting system developers themselves can place malicious applications and deliver the product (i.e., new or modified e-voting systems) to election authorities (Wolf et al., 2011).

Lack of public confidence. Some people do not trust e-voting systems due to several reasons. They believe authorized election personnel or hackers can alter their ballots easily. Due to the lack of public confidence, some countries like the UK and the Netherland have already moved away from using e-voting systems in their local and national elections (Esteve et al., 2012).

Main Security Areas of E-Voting

As critical software systems, electronic voting systems must satisfy several important security requirements, such as confidentiality, integrity, availability, and privacy.

Confidentiality. Confidentially refers to safeguarding a voter's sensitive information (i.e., ballot data, voters' information, etc.) and other data related to the system from being disclosed to unauthorized persons or parties. In other words, it must properly protects the secrecy of the ballots (Chia, 2012).

Integrity. Integrity means safeguarding the accuracy of ballot data and software integrity of the system. All these sensitive data must be protected from unauthorized modifications. The trustworthiness of both the data and the system functionalities plays a very important role in e-voting systems (Chia, 2012).

Availability. Availability refers to the amount of time that the system is accessible to the users (voters and all other election authorized personnel and systems). The system may still be available even under reduced speed. However, due to some external interference, such as malicious attacks or any other system malfunction, it could become unavailable (Chia, 2012).

Privacy. Privacy in e-voting means, anyone should not be able to associate a ballot with the voter who casted it. A voter should not be able to prove how he/she cased his/her vote to an external party, so that vote buying and extortion can be prevented. When using e-voting systems, some voters may face difficulties and may need assistant from another person due to some disabilities, such as partial blindness. If that occurs, privacy of those voters might be violated when a second person interferes. In e-voting, election administrators or other authorized personnel may not need privacy protection as do the voters in an election (Cranor & Cytron, 1996).

Other Properties of E-Voting

Receipt freeness. This property helps to prevent a voter to prove to another person or any other party, how he or she casted his or her vote (Delaune, Kremer, & Ryan, 2006).

Non-repudiation. This property prevents the actual sender of data (e-ballot etc.) from denying they are the actual sender (Rusinek & Ksiezopolski, 2009).

Literature Related to the Problem

Santin, Costa, and Maziero (2008) presented a three-ballot-based secure electronic voting system which is integrated into a single architecture. The system considered securing voter privacy, anonymity, and vote receipts, etc. The proposed system aims to satisfy security requirements of e-voting systems by reducing the complexity by using classic cryptography techniques such as standard public key cryptosystem rather using visual cryptography. This practical approach was implemented as a prototype by mean of election markup language (EML) and other web services. The proposed approach could be applied in an election which covers a large geographical area, as well as in several other elections such as corporate and academic elections while reducing the cost of deployment.

Ansari et al. (2008) stated that government agencies have been replacing traditional paper-based voting systems with electronic voting systems. Although the electronic voting system are certified by federal and state government agencies, many people question their privacy, security, and performance. This paper presented the findings of a project that revealed several threats against the security requirements of electronic voting systems such as violation of voters' privacy. Moreover, the authors of this paper suggested several solutions for the development of electronic voting systems to minimize the threats against security requirements, such as developing direct record election (DRE) systems, in a way where these systems can hide the voter selection screen from election officials when the voters seek assistance during the election process.

Sebe, Miret, Pujolis, and Puiggali (2010) discussed the importance and current challenges of electronic voting such as ensuring security requirements of remote e-voting systems. This study mainly discussed the problems of using complex and costly approaches in verifying the correctness of voting process, particularly in current vote mixing verification processes of remote e-voting systems. The authors of this paper proposed a new remote voting scheme which made the voting process more efficient, less complicated, and cost-effective while satisfying the security requirements of electronic voting such as integrity, confidentiality, and privacy.

Olaniyi, Arulogun, Omidiora, and Oludotun (2013) designed a secure voting system primarily focused on improving the security requirements for authentication and integrity of e-voting systems in an efficient and reliable way. The authors of this paper highlighted the important characteristics of ballots in a democratic election such as anonymity and tamper resistance of ballots. The proposed design of the secure voting system used cryptographic hash functions to assure the integrity and one time short message service (OTSMS) plus grid card multifactor authentication to minimize possible errors that may take place during voter authentication.

Kumar and Srinivasan (2013) proposed a practical approached to preserve privacy of electronic voting. The authors discussed three types of internet voting systems, namely poll site, kiosk, and remote and also explained the advantages and the problems of those e-voting systems. The proposed new scheme used smart card techniques which employed blind signature. The design of the new system focused on key functions of the new generation smart card technologies that aided to secure the operations of internet voting systems and the privacy of the voters.

Literature Related to the Methodology

In XML, there are three types of digital signatures as follows (Bartel, Boyer, Fox, LaMacchia, and Simon, 2013; XML digital signature [APA], n.d.).

Enveloped. The signature element is placed in the XML document as a child element of the root element of the XML document.

```
<election>
<vote>
      <chairperson> candidate1 </chairperson>
      <secretary> candidate2 </secretary>
</vote>
<signature>
         . . . . .
</signature>
</election>
```

Enveloping. Here the signature element is the root element of the XML document and the main document elements become the child elements of the signature element. The actual data elements are embedded in an auxiliary tag such as object tag.

```
<signature>
:
<object Id = “something”>
<election>
<vote>
                 <chairperson> candidate1 </chairperson>
                 <secretary> candidate2 </secretary>
         </vote>
  </election>
</object >
</signature>
```

Detached. In here, neither enveloped nor enveloping is used. The signature elements and data can be in the same XML document or in a separate file. When these two in the same document, signature and data become siblings.

```
<rootelemet>
        <election>
                <vote>
                        <chairperson> candidate1 </chairperson>
                        <secretary> candidate2 </secretary>
                </vote>
        </election>

        <signature>
         :
        </signature>
</rootelemet>
```

The signing process consists of three steps.

i. The signed electronic document is canonicalized by using C14N algorithm. By doing this, impact on the signature from different formatting can be avoided

ii. Document hash value or digest is computed using hash algorithm such as SHA1, SHA256 or MD5

iii. The signature is encrypted using the private key of the sender. Public key algorithm such as RSA DSA is used for this purpose.

During the validation receiver perform the following steps.

i. The XML document is canonicalized

ii. The signature is decrypted using sender’s public key and then recomputed the hash value

iii. If the comparison of recomputed hash and the declared hash value pass, then the validation passes.

Figure 3 depicts how an electronic document is digitally signed and Figure 4 depicts how to verify the integrity of the document.

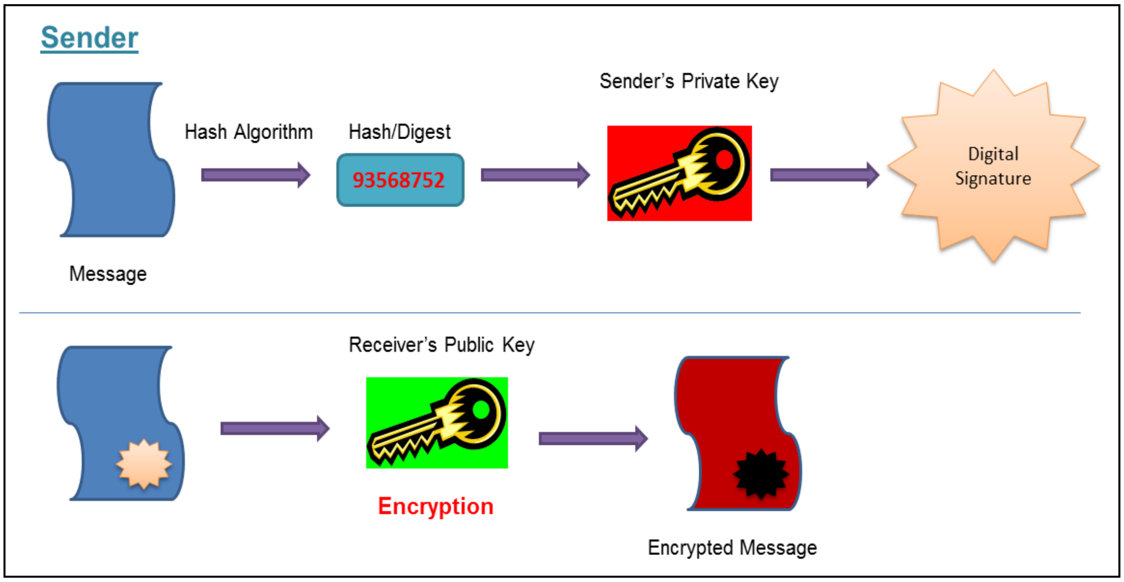


Figure 3. Digital signature signing.

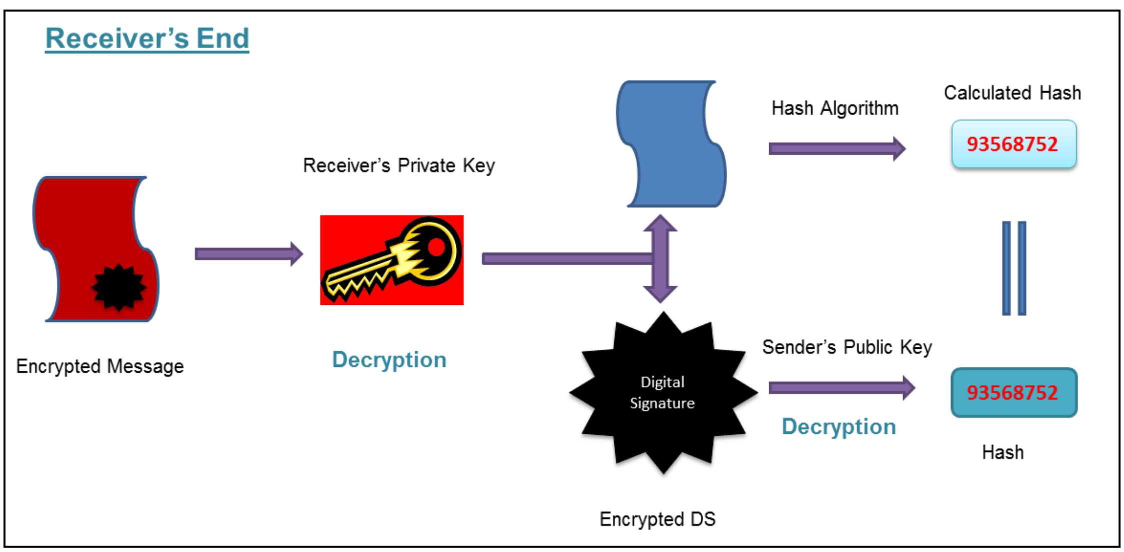


Figure 4. Digital signature verification.

The signature element. The following structure shows the simplest form of XMLDSIG element which is a key element of the proposed system. <SignedInfo> element contains information regarding the signature.

```
<Signature>
<SignedInfo>
    (Canonicalization Method)
    (Signature Method)
    <Reference>
            (Digest Method)
            (Digest Value)
    </Reference>
</SignedInfo>
(Signature Value)
</Signature>
```

Symmetric key cryptography. This is also called secret key algorithm. For both encryption and decryption, same key is used. Therefore, the key has to be delivered securely to the other party. Compared to asymmetric cryptography, symmetric cryptography is comparatively fast. Advanced encryption standard (AES) is one of popular symmetric key algorithm (Pfleeger & Pfleeger, 2007).

Asymmetric key cryptography. This is also known as “Public key cryptography”. A pair of independent keys is (Private/Public) used to encrypt and decrypt data. Although the public key is publicly known, determining corresponding private key is almost impossible. With Public key cryptography, no need to worry about secure exchange of keys among all involved parties. However, this method is slower than Symmetric key cryptography and also requires more computer processing power for both encryption and decryption. The RSA algorithm is an example for public key cryptography algorithms (Konheim, 2007; Whitman & Mattord, 2009).

Hashing algorithms. Hash algorithms are used to generate a value (this value is also known as a digest) according to its input data, such as a message or a document. When data are sent over unsecured channels, hash values are used to verify the integrity of data (Ciampa, 2009). These algorithms can be easily used to defeat Man-in-the-middle attacks. SHA-1 is a popular hash algorithm.

Summary

In this chapter, background and literature related to the problem as well as literature related to the methodology were discussed. The next chapter explains in detail the methodology that was followed in this thesis.

Chapter III: Methodology

Introduction

In this chapter, the design of the proposed e-voting system is explained in detail. The steps of the new system's design structure, the use of cryptographic techniques in the system design, and the use of XML-based technologies in the electronic voting processes, are covered in detail in this chapter.

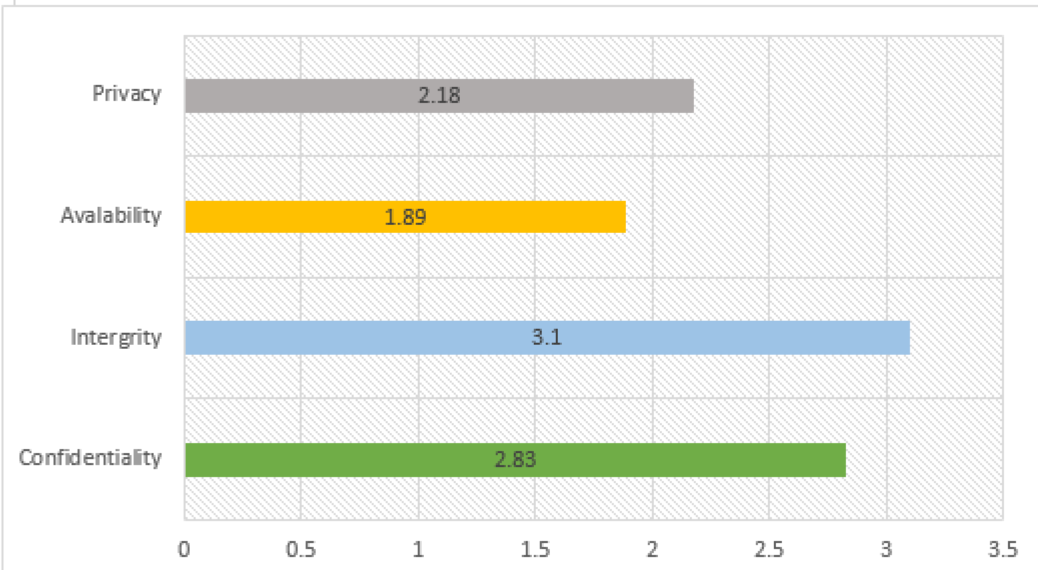
Survey of E-Voting Systems

In this thesis, before developing the proposed new e-voting system, a survey was conducted regarding the security requirements of e-voting systems. The survey results were helpful to identify the most important security attributes and major security concerns of current e-voting systems. When replacing traditional voting procedures with e-voting, people will expect the same properties as with a paper ballot system. Voters' point of view, lack of security on integrity and privacy can mainly contribute to deprivation of voter's rights. Even software bugs are a threat to software integrity while data integrity can be compromised through unauthorized modifications. However, this thesis only concerns issues related to the data integrity rather than software integrity of the e-voting system.

Every e-voting system must take strong steps to avoid compromising voters' privacy. Otherwise, they will have to face serious political, social, legal, and ethical problems. These negative outcomes may include even financial losses or damaged reputation. The following survey results show that most of voters are highly concerned about the integrity and the privacy of their votes. In addition, the survey shows that the potential threats for e-voting systems in terms of integrity and privacy are also high. Complete survey questions and their answers are shown in Appendix A.

Q4: How do you rank the following e-voting attributes according to potential threat level (Give one to highest)?

Figure 5 visualizes the results of the survey for this question.

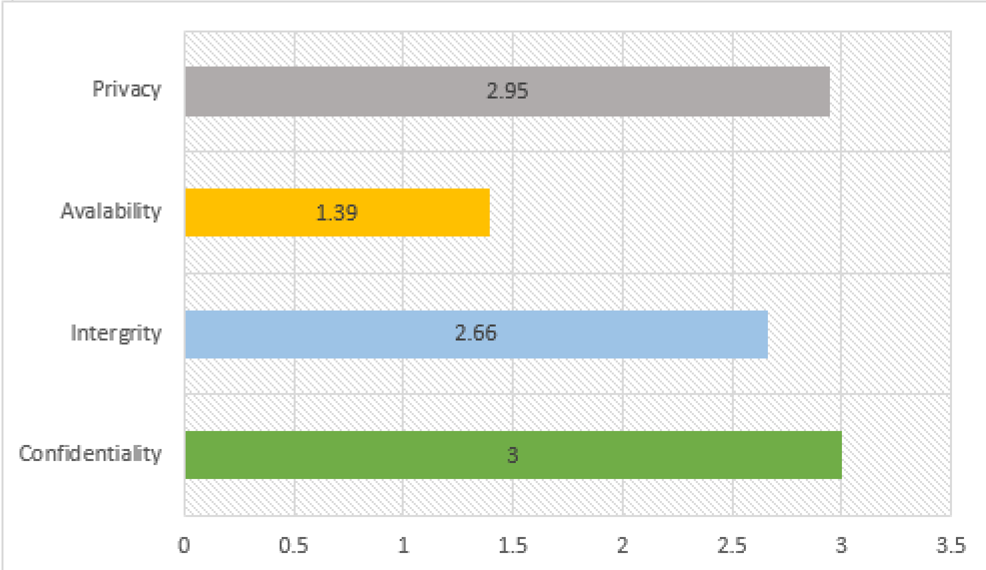


	1	2	3	4	Total	Score
Confidentiality	21	46	28	5	100	2.83
Intergrity	46	22	28	4	100	3.1
Avalability	16	6	29	49	100	1.89
Privacy	17	26	15	42	100	2.18

Figure 5. Survey question 4.

Q5: How do you rank the following e-voting attributes according to the level of harmfulness on voters if a malfunction take place in voting process (Give “one” to the most harmful attribute)?

Figure 6 visualizes the results of the survey for this question.

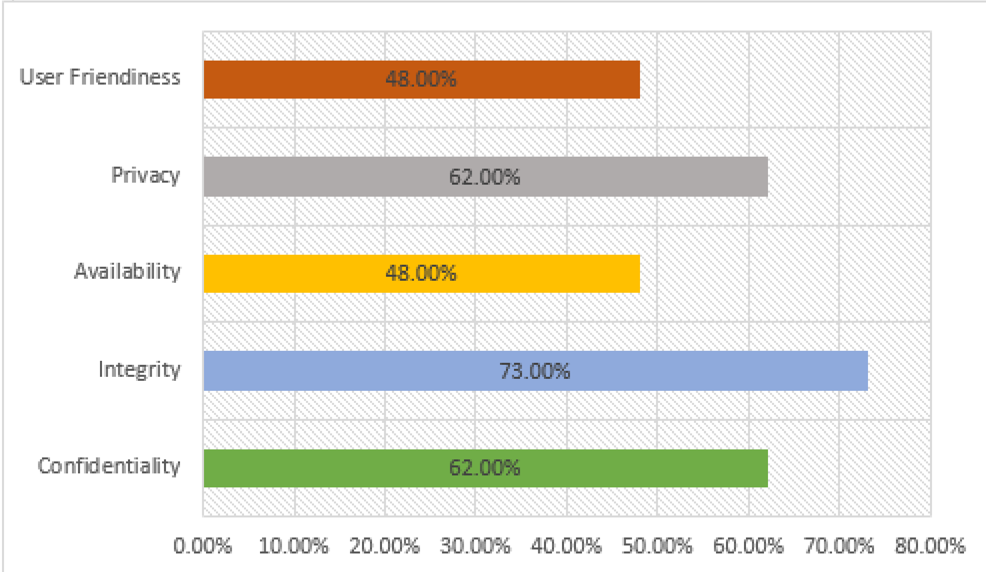


	1	2	3	4	Total	Score
Confidentiality	27	49	21	3	100	3
Intergrity	24	20	54	2	100	2.66
Avalability	6	3	15	76	100	1.39
Privacy	43	27	10	19	100	2.95

Figure 6. Survey question 5.

Q6: Select your top 3 e-voting attributes that you expect from an e-voting system as a voter.

Figure 7 visualizes the results of the survey for this question.

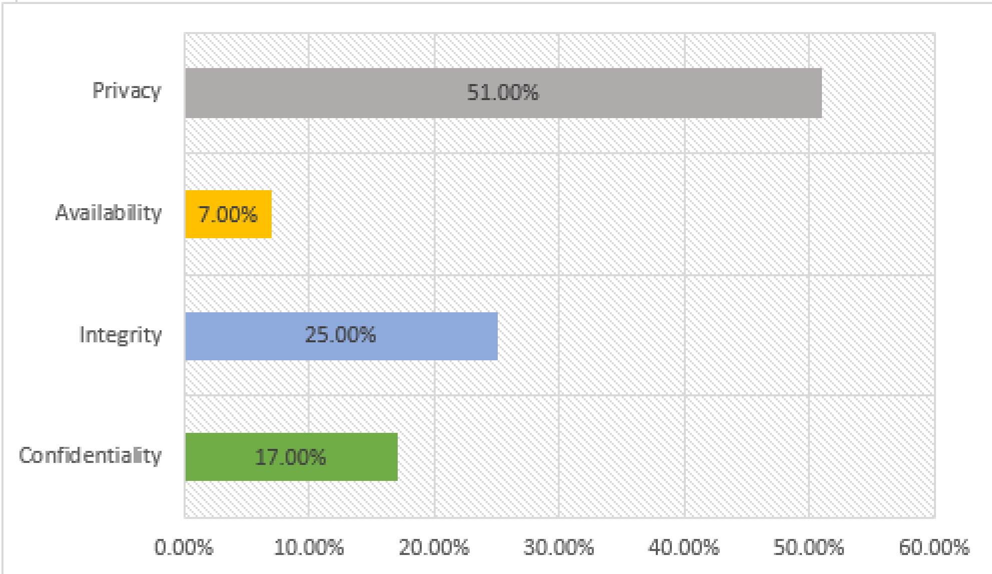


Answer Choices	Responses	
Confidentiality	62.00%	62
Integrity	73.00%	73
Availability	48.00%	48
Privacy	62.00%	62
User Friendliness	48.00%	48
Total Respondents: 100		

Figure 7. Survey question 6.

Q8: In your opinion what is the e-voting attribute which has more social and/or ethical issues?

Figure 8 visualizes the results of the survey for this question.



Answer Choices	Responses	
Confidentiality	17.00%	17
Integrity	25.00%	25
Availability	7.00%	7
Privacy	51.00%	51
Total		**100**

Figure 8. Survey question 8.

Design of the Study

The design of the new prototype e-voting system accomplishes the following tasks.

i. Creates separate electronic ballot (e-ballot) for each voter very efficiently

ii. Assures the integrity of each ballot by means of XML Digital Signature

iii. Assures the privacy of the voter by means of XML Encryption Technologies and new design architecture.

Because the major goal of this thesis is to assure the integrity and the privacy of EVS used in small and mid-size electoral populations in an efficient and cost-effective way, the major processes, such as encryption mechanism and ballot singing process should be completed very quickly and in a less expensive manner, so that nobody or any malicious applications such as Trojans or other virus applications have sufficient time to perform their malicious activates. In computer-based systems, minor process delays may create unprotected and unintentional entry points to breach the security of the system. Therefore, introducing lightning fast data processing is very important aspect of security of critical systems such as EVS.

Therefore, creating a separate ballot for each voter would be an ideal solution to make the whole process efficient by reducing the amount of data to be processed during the election phase. Apart from that, individual ballot approach helps to secure voter's privacy better than storing all voters' ballot data in a single file, in case the system's security is breached. The single file method may make all data vulnerable to malicious activities and might not give sufficient time to isolate unaffected data from infected data, after detecting any ongoing unauthorized activities. Thus, individual ballot method can be used to isolate tampered ballots easily.

In order to assure the integrity, ballot data need to be secured throughout the election process. The robust XML Encryption technologies (Imamura et al., 2013) come in handy to achieve this goal when XML data represent the ballot data (Al-Hamdani, 2010; Imamura, Clark, & Maruyama, 2002). However, as the second step of integrity assurance procedure, tampering of data by any means need to be detected. At this point XML Digital signature plays an important role in detecting any data modifications after the election process (The admin process signs each ballot immediately after voter cast his/her vote) (Dournaee, 2002; Eastlake & Niles, 2002). The steps of the proposed e-voting system's processes are as follows.

i. Voter cast his/her vote using a DRE kiosk or through a web interface
ii. An electronic ballot (e-ballot) is created for each voter separately and all required ballot data are stored in the e-ballot, which is in XML format. The data is stored in XML elements in a way it supports to secure the privacy of voters
iii. Each ballot is directed to the election administration process. The administration process creates copies of the e-ballot for each process of the electronic voting system
iv. Digest is generated for each process. Thereafter, separate digital signatures for each process are embedded into the e-ballots
v. Data belonging to each election process is encrypted through XML encryption to safeguard integrity
vi. Upon receiving e-ballots, each process decrypts the data belonging to them. The digital signature for the process is now accessible
vii. Each process verifies the integrity of e-ballots by means of a separate public key given by the administration process to each election process

viii. Tally process can verify voters' identification (ID) by sending encrypted IDs to the administration process without exposing sensitive information

ix. Tally process starts counting votes in the system.

The following paragraphs explain in detail the whole process of the proposed e-voting system. Figure 9 shows the overview of the proposed system.

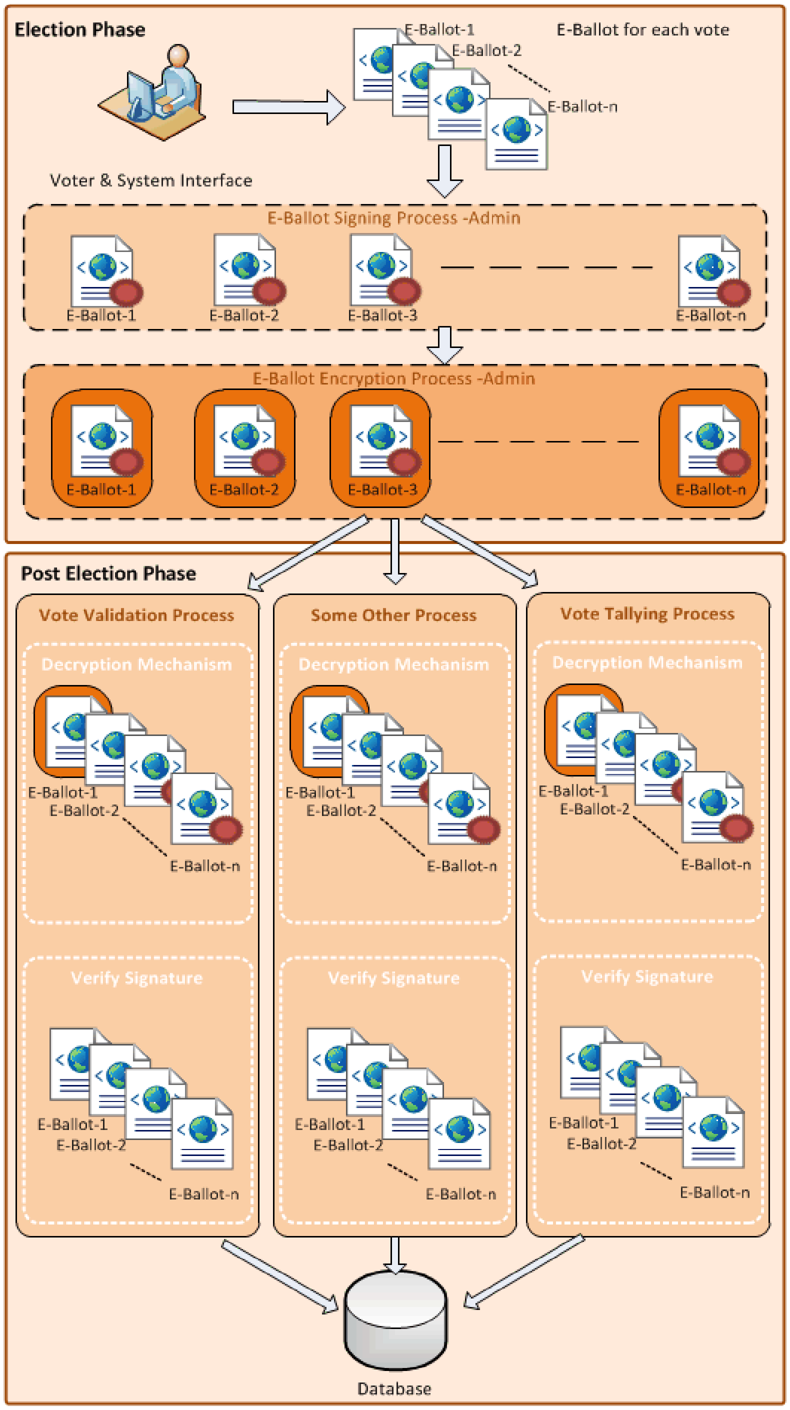


Figure 9. Overview of the proposed e-voting system design.

As the starting point when the user logs on to the proposed e-voting system, the voter can make his/her choices and cast the vote. The moment he/she presses the "Press Here to Vote" button, an e-ballot is created in XML data format and stores all required data in pre-defined elements in the XML document as shown in Figure 10. Each e-ballot represents the vote of only one voter.

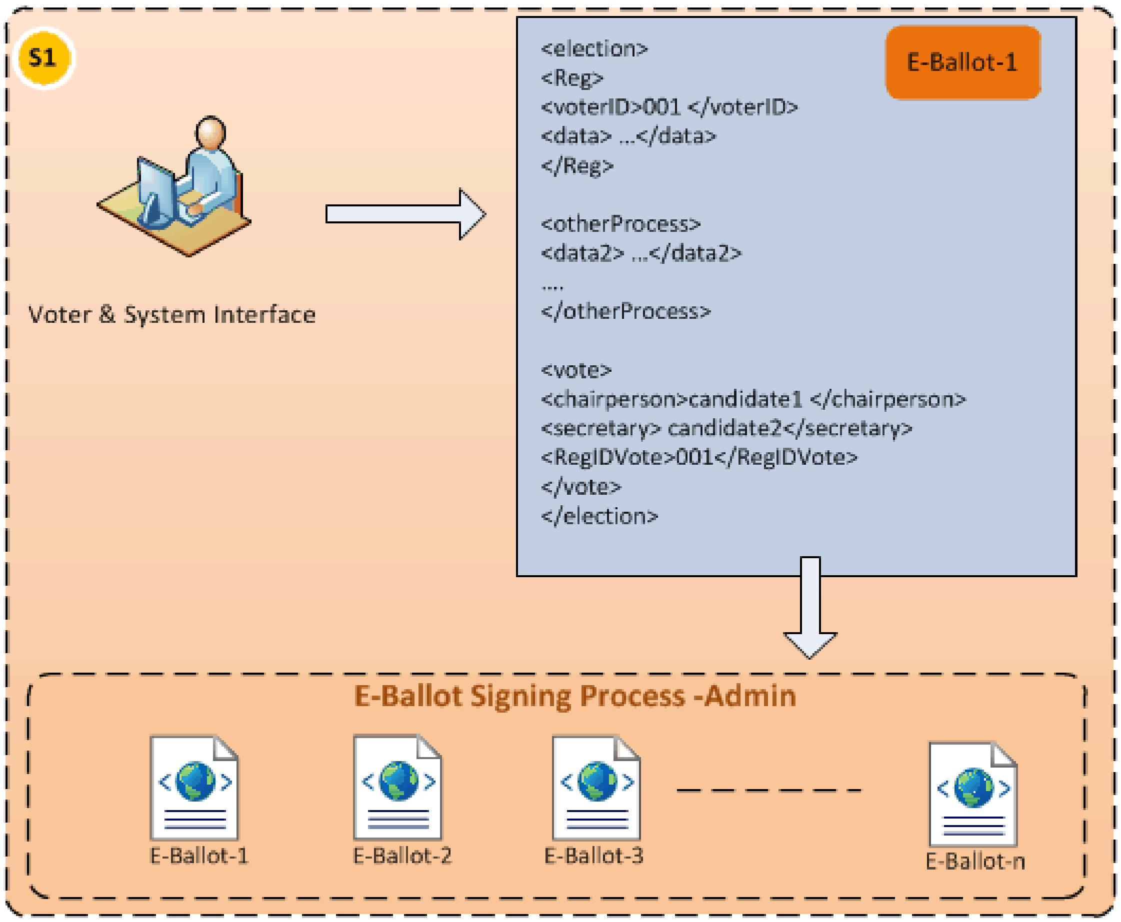


Figure 10. Voter cast ballot.

The data processing starts at the administration process that generates digests from the data in the e-ballots. Once it finish generating the digests using SHA-1 hash function, digital signatures for each process are generated with the private key of the administration process as shown in Figure 11. For instance, if the e-voting system has three other processes in addition to administration process, the administration process generates three different digital signatures for each process and embeds those digital signatures in the e-ballot. The other three processes will have the relevant public key for the decryption process.

Even though it is almost impossible to determine the private key from the public key, in this proposed system, three different private/public key pairs are created for the digital signature processes of the three processes in order to further increase the security of the system. However, because only the digests are signed instead of the entire document data, computer processing will be significantly reduced and signing process will be performed promptly (Digital Signatures [APA], n.d.).

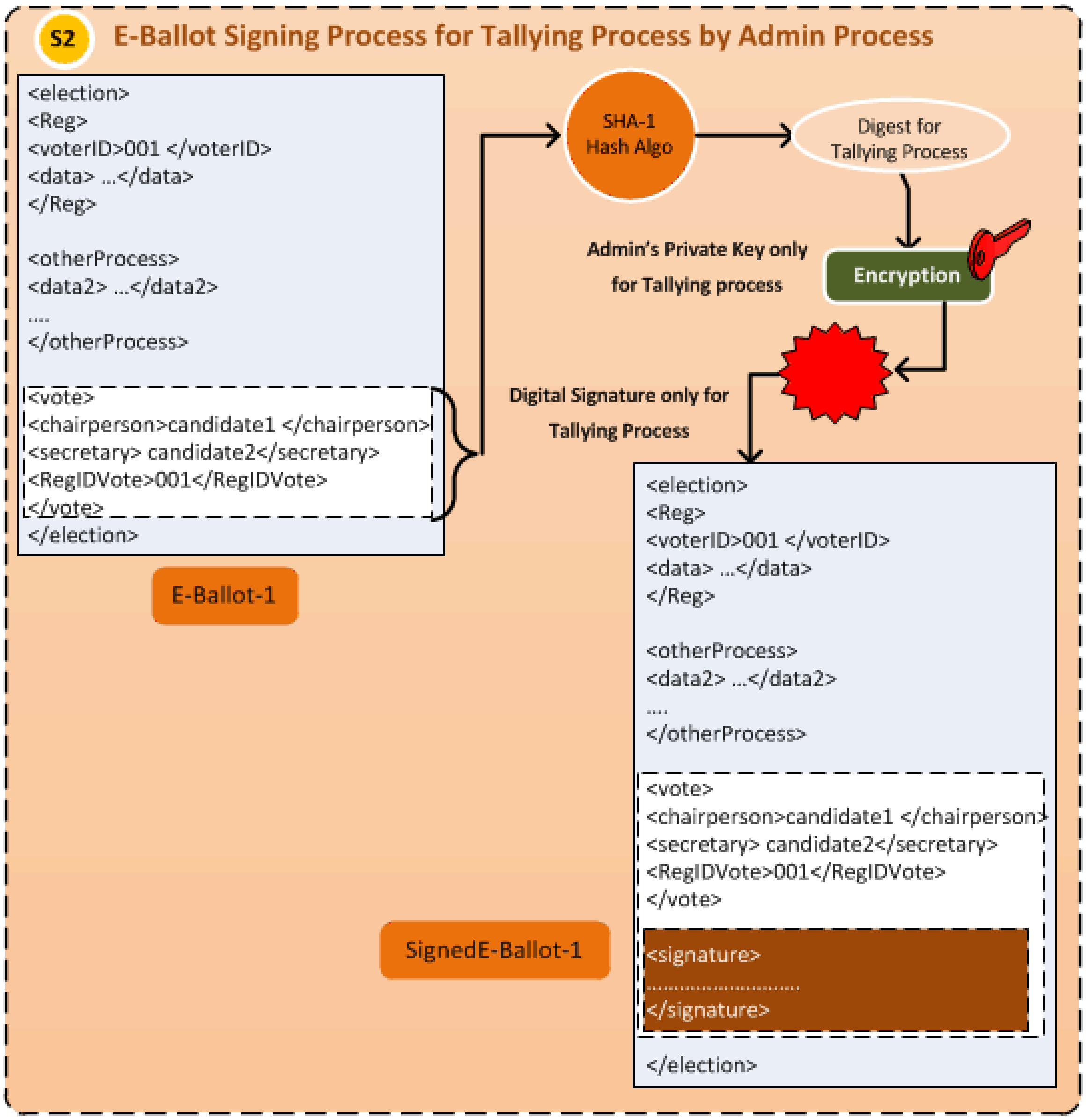


Figure 11. E-ballot signing process.

Here, RSA algorithm is employed as the asymmetric key algorithm. In order to secure the integrity of data, administration process encrypts the relevant data of each process with the public key of appropriate process. For instance, data belong to tally process is encrypted by the public key of the tally process as shown in Figure 12.

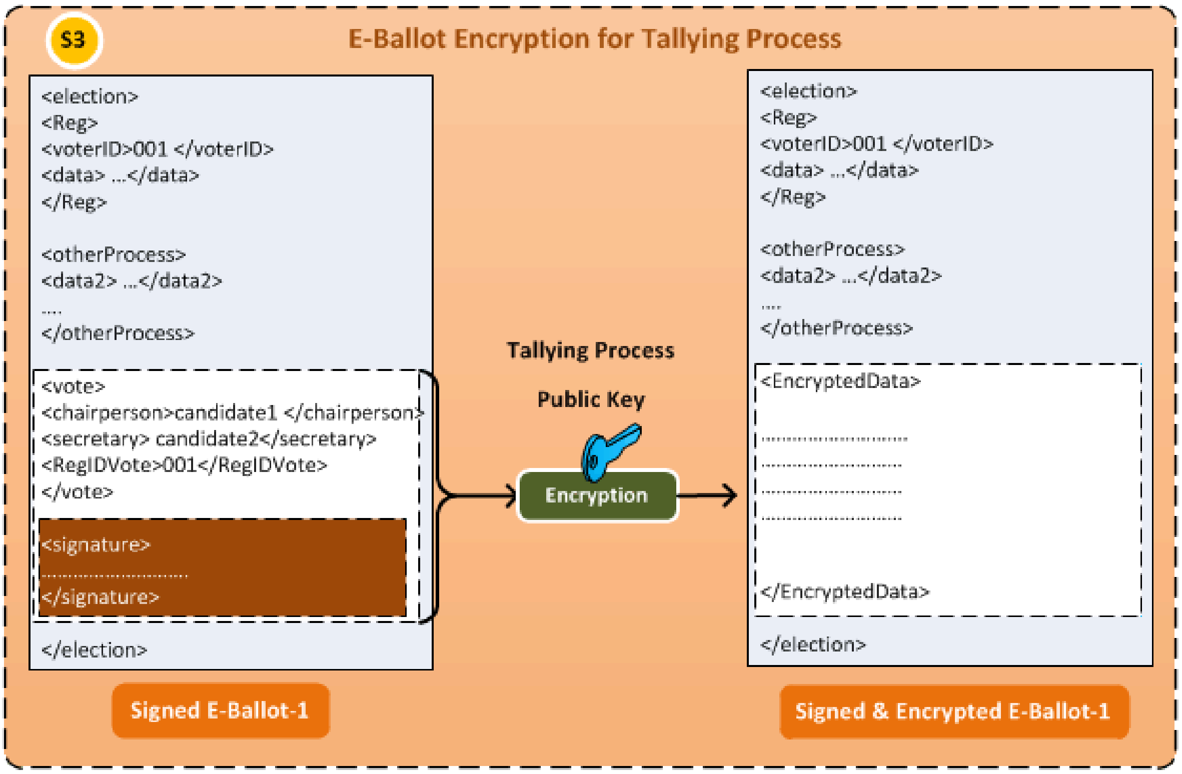


Figure 12. E-ballot encryption.

Once the election phase is over, all ballots (i.e., encrypted and digitally signed ballots) are sent to each process by the administration process as shown in Figure 13. Upon receiving ballots, each process decrypts the ballot data by means of their private keys and prepares ballots for the validation.

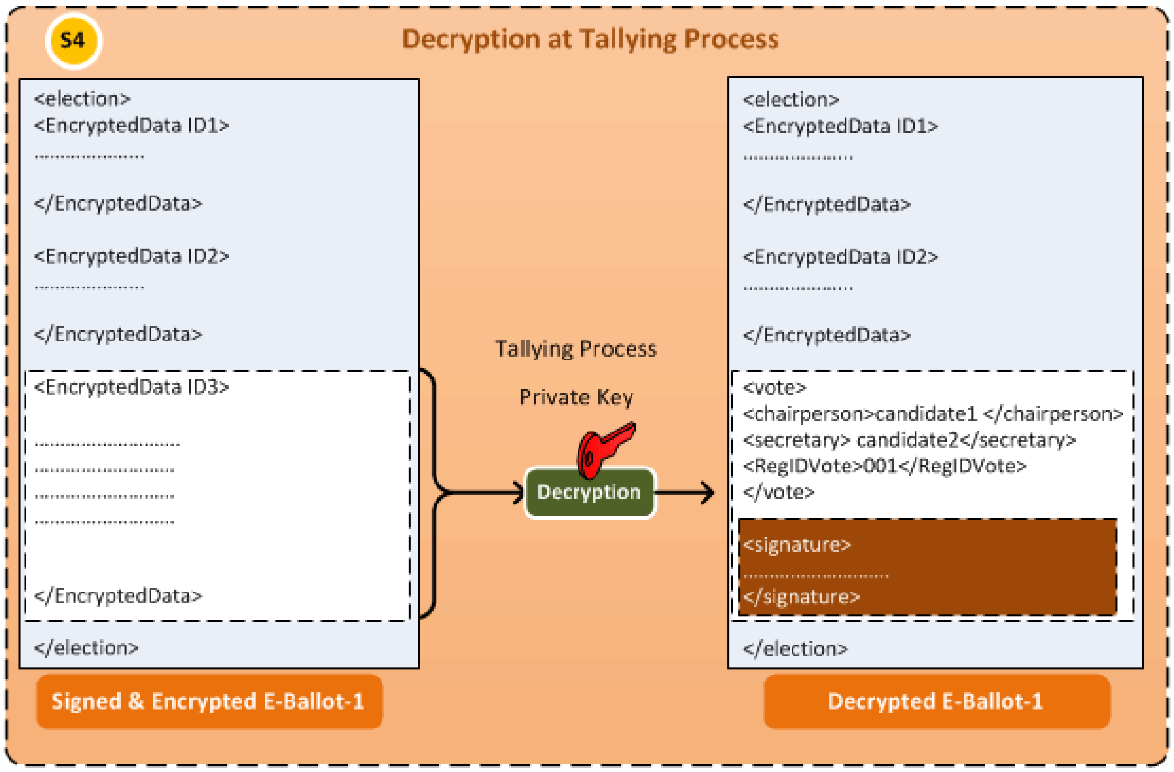


Figure 13. E-ballot decryption.

Signature is obtained by using admin’s public key that generated only for the tally process and the digest is regenerated using the same algorithm which used at the second step. If the previous digest and the regenerated digest are not the same, it implies that unauthorized person or system tampered with the data. Figure 14 depicts this signature verification process.

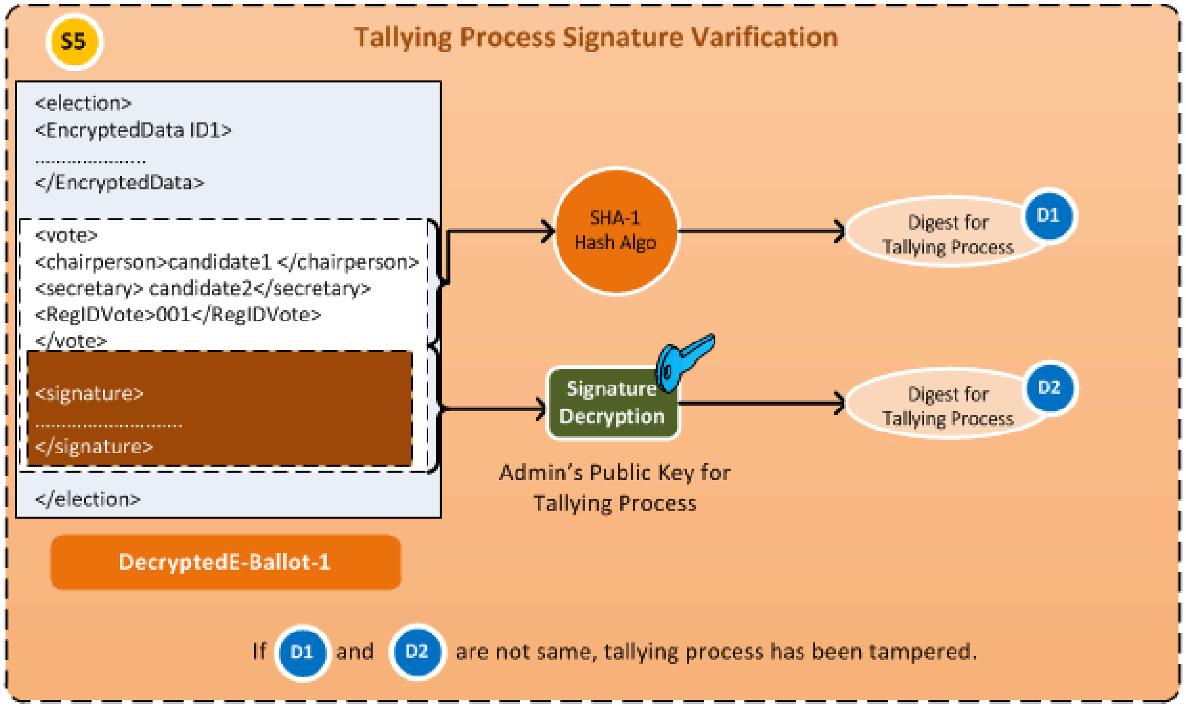


Figure 14. Signature verification.

A person (i.e., authorized personnel in the election process or any other outside person) or an unauthorized process or a third party malicious application must not be able to map voters to their ballots. The structure of the elements in the e-ballot is defined to ensure the privacy of voters. Some processes might need to have voter's ID within their authorized domain for an activity like "Voter Verification". At this point, administration process encrypts the ID with the administrator's symmetric key and places the encrypted voter's ID within the area belonging to the specific process as shown in Figure 15. For this purpose the advanced encryption standard (AES) algorithm will be used by the administrator.

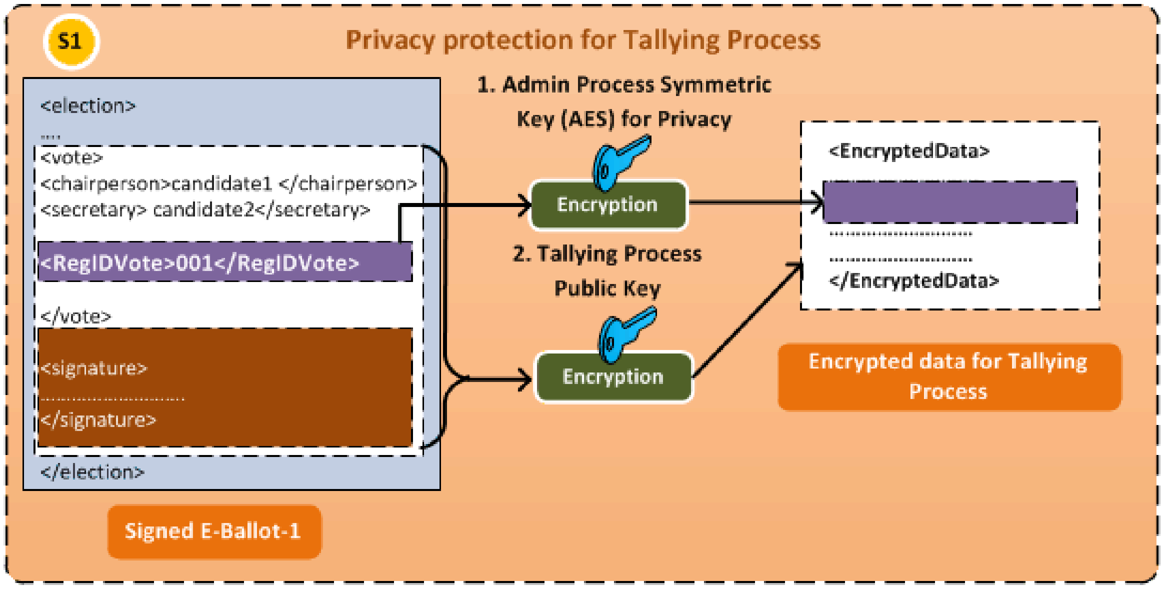


Figure 15. Privacy protection.

One can imagine the following scenario to get an idea about how to protect the privacy of the voter from being breached. Assume that the tally process needs to verify the voter’s ID before starting the counting sub-process. The voters’ ID in the e-ballots are encrypted and the tally process cannot decrypt and see the voters’ Ids in plain text. At this time, the tally process can only request the administration process to validate voter’s IDs available within its domain. Then, the administration process validates each ID and issues the status of the ID to the tally process. The tally process can then tabulate their data as shown in the following table.

Table 1

Tally Process Data

Voter's ID (Scrambled data)	ID Status	Vote					
		Chair Person			Secretary		
		Candidate1	Candidate2	Candidate3	Candidate1	Candidate2	Candidate3
wetegertgerttqw	Valid	X			X		
tettegfhtyjkuywe	Valid		X				X
Ytiutyityitiiu56trt	Invalid			X	X		
teryrhbnfkujywe	Valid		X			X	

The tally process or admin process will never see a voter's ID and candidate selections of the voter together to breach privacy. Figure 16 depicts how this step is performed by the system. After successfully performing all tasks including functionalities related to integrity and privacy, ballots are stored in the secure database.

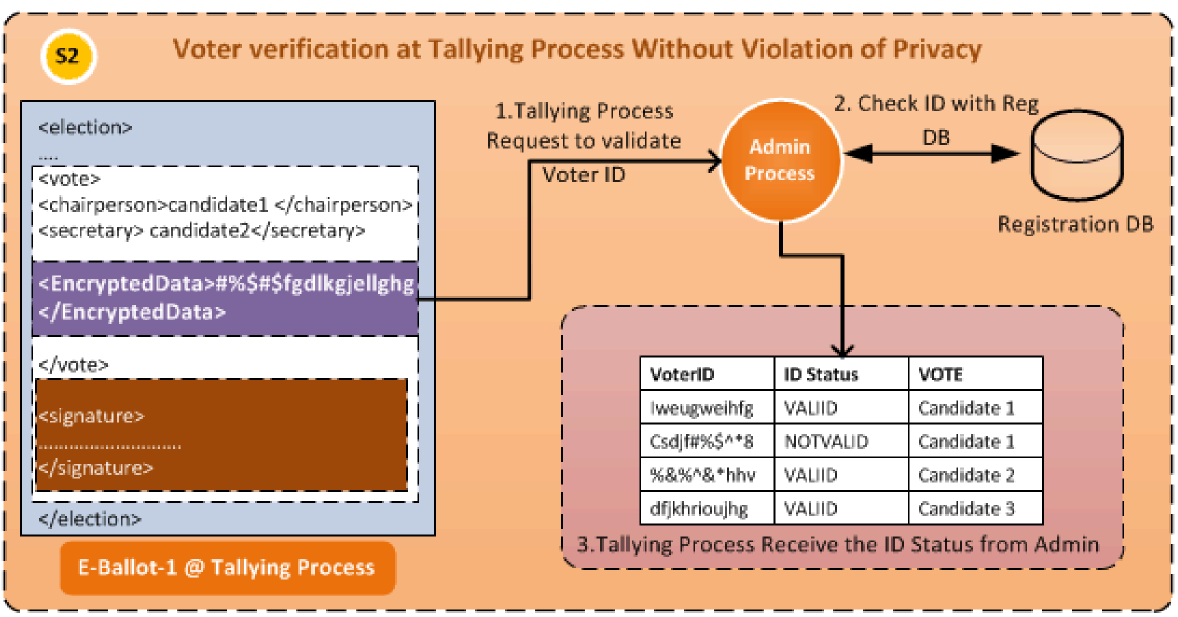


Figure 16. Voter verification.

Because the administration process generate separate digital signatures for each process, each process can behave independently and verify the integrity of the receiving ballots by their own. Otherwise, one process needs to depend on another process to obtain the status of the integrity of the e-ballots. That kind of dependent approach may introduce some security flaws during the election operation time.

For instance, if the validation process is responsible for validating the digital signature and the validation process is compromised by an unauthorized party, then the unauthorized party can issue a false status of integrity to the other dependent processes. Even in a situation where the validation process is not compromised, a malicious program would be is capable of executing its malicious operations in between two processes, alter the status of the integrity, and make the other process believes that the e-ballots have not been tampered with. In e-voting systems, many malicious attacks, such as man-in-the-middle attacks which are launched by hackers, can be avoided successfully by means of XML digital signature technologies (Ciampa, 2009). However, for various other purposes each process needs to collaborate with other process. The above mentioned independency is important, because the integrity of the ballots is very crucial in terms of the validity of the whole election.

Summary

This chapter explained how the new proposed e-voting system was designed in order to secure its integrity and privacy by using cryptographic and XML based technologies. The next chapter explains in detail how the proposed approach is implemented on a Windows platform as a prototype application.

Chapter IV: Implementation

Introduction

In this chapter, the implementation of the proposed EVS is explained. The new EVS was named as X-Ballot and it followed the methodology explained in Chapter III. This chapter explains in detail the technologies, techniques, and software tools used to implement the X-Ballot system. Figures of both system user interfaces and system generated files were used to illustrate how the proposed new X-Ballot system works.

System Implementation

The basic architecture of the X-Ballot system is based on the direct recording electronic (DRE) voting system and the X-Ballot employees Microsoft Windows based XML encryption and digital signature technologies for the implementation.

The X-Ballot was implemented with Microsoft Visual Studio.Net 2013 framework (How to: Sign XML Documents with Digital Signatures [APA], n.d.; How to: Verify the digital signatures of XML documents [APA], n.d.) using C# programming language. The Visual Studio framework supports XML security standards used in the proposed approach and the .Net framework produced the required infrastructures such as library classes for encryption/decryption of XML documents and other essential classes for digital signature process. Furthermore, graphical user interfaces (GUI) of the X-Ballot were also created using Visual Studio.Net 2013.

In order to begin the voting process, authorized users (voters) can log into the system using valid user credentials that are provided by the administration of the election during the voter registration process. Voters can use the vote form shown in

Figure 17 to cast the electronic votes. Voter’s ID information is displayed at the top of the form (i.e., voter IDs’ for general registration process and voter’s ID only for the tally process). Using this vote form, voters are able to select one candidate for each job post (i.e., President, Secretory, and Treasurer) shown in the form. The voter’s selections are displayed at the bottom of the form so that the voter can make sure as to whom they are going to vote for. After the confirmation, the voter can press the “Press Here to Vote” button at the bottom of the vote form to cast his/her vote.

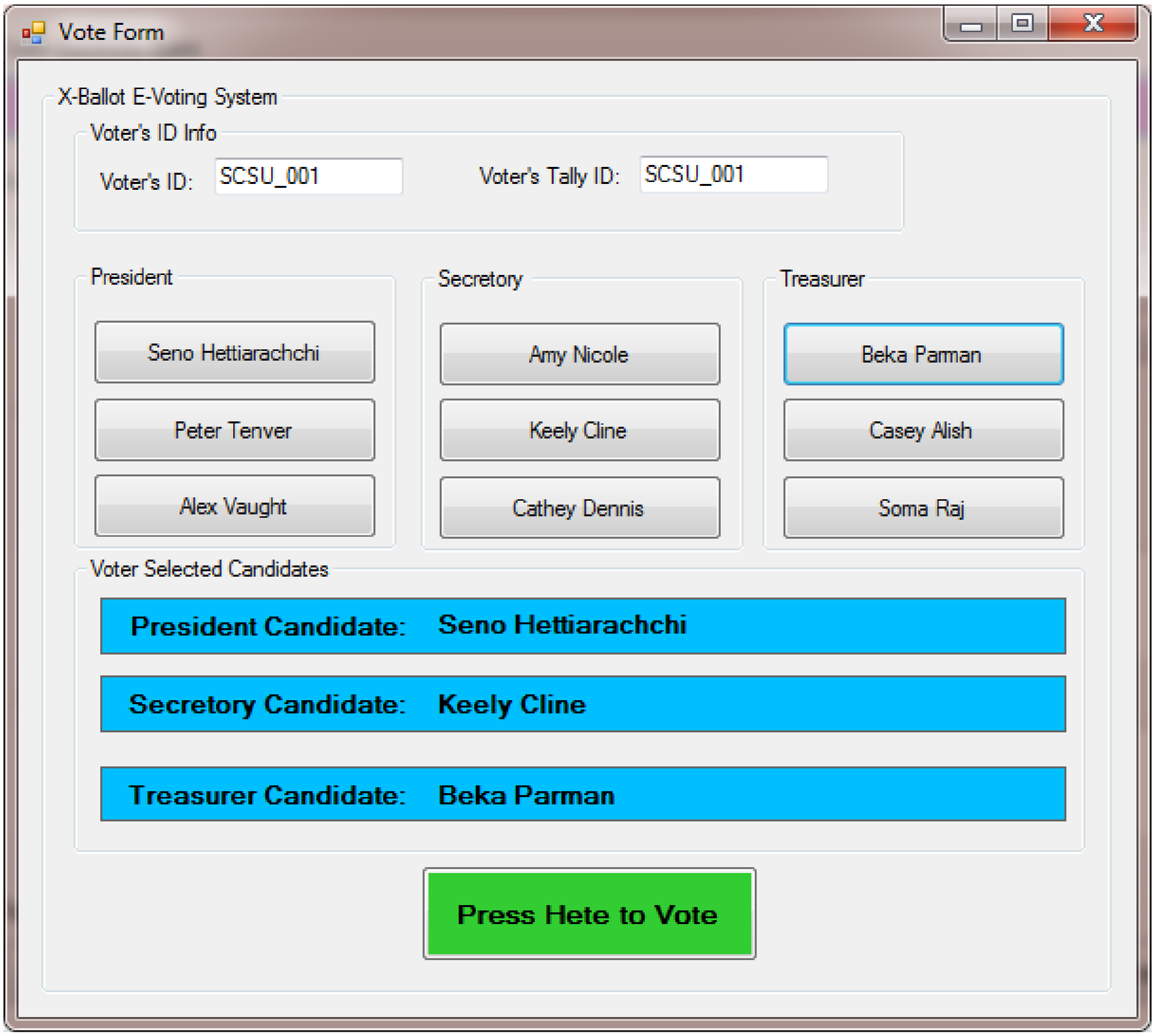


Figure 17. Form to cast ballot.

When the voter press the "Press Here to Vote" button, the X-Ballot system creates an Xml file (i.e., eballot-x.xml) in which voter's all ballot data are stored. Note that, "x" represents 1, 2, 3, and so on. For instance, 1st electronic ballot is named as eballot-1.xml. The X-Ballot system generate individual electronic ballot in a form of xml for each and every voter who cast their vote using the X-Ballot system.

The X-Ballot uses the XDocument class to efficiently create eballot-x.xml files rather than using XmlWriter class (XDocument Class Overview (C#) [APA], n.d.). After creating an eballot-x.xml file, copies of the eballot-x.xml files are sent to all processes of the X-Ballot system with encrypted data. Figure 18 shows a sample code of the eballot-x.xml file that utilize the XDocument class in the X-Ballot system.

```
XDocument xdoc = new XDocument(
        new XDeclaration("1.0", "utf-8", "yes"),
        new XElement("eballot",
              new XElement("selections", new XAttribute("id", "t"),
                  new XElement("president", lblDispalyPre.Text),
            ⋮
            ⋮
xdoc.Save ("eballot-x.xml");
```

Figure 18. E-ballot creation with required elements.

Figure 19 shows a sample eballot-x.xml file with election data (i.e., chosen candidates, voter’s ID, etc.). This sample file displays data in plain text (unencrypted data) and the structure of the xml document.

```
  <?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <eballot>
  - <selections id="t">
        <president>Ranolee Hettiarachchi</president>
        <secretory>John Kinght</secretory>
        <treasurer>Bob Steve</treasurer>
      - <sel_reg>
            <voter_reg_id>SCSU_001</voter_reg_id>
        </sel_reg>
    </selections>
  - <regprocess>
        <voter_reg_id_ori>SCSU_001</voter_reg_id_ori>
    </regprocess>
  - <valprocess>
        <val_data>4</val_data>
    </valprocess>
  - <admprocess>
        <election_data>5</election_data>
        <election_data>6</election_data>
    </admprocess>
  </eballot>
```


Figure 19. eballot-x.xml file.

Immediately after casting the vote, the eballot-x.xml file is encrypted by the election process’s administrator and send copies of the files to each process in the X-Ballot system. Figure 20 shows the main administration form used in this prototype application. The next steps of the X-Ballot system are explained through the tally process.

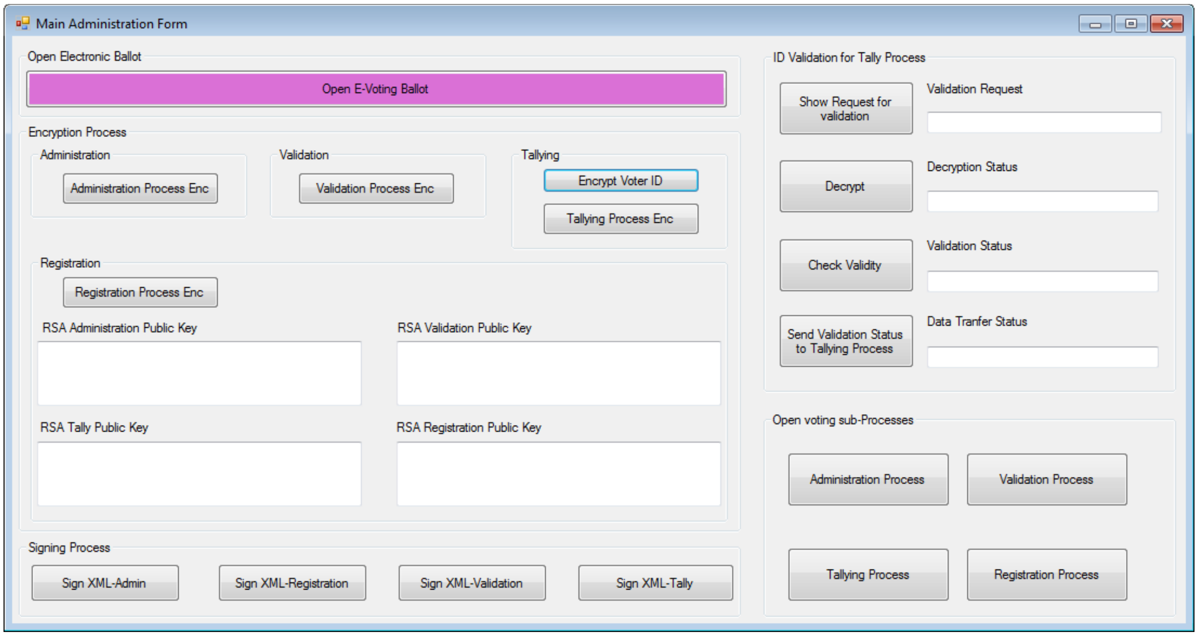


Figure 20. Main administration form.

Before sending the eballot-x.xml file to the tally process, the <voter_reg_id> is encrypted with election process administrator's asymmetric key for minimizing privacy related matters which will be discussed in detail later in this chapter. Figure 21 shows the eballot-x.xml file after encrypting the <voter_reg_id> element.

```xml
  <?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <eballot>
  - <selections id="t">
      <president>Ranolee Hettiarachchi</president>
      <secretory>John Kinght</secretory>
      <treasurer>Bob Steve</treasurer>
    - <sel_reg>
      - <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
        Type="http://www.w3.org/2001/04/xmlenc#Element" Id="EncryptedElement20">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
        - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
          - <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
              <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
            - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                <KeyName>pri_key</KeyName>
              </KeyInfo>
            - <CipherData>
                <CipherValue>RW7F3gJkGgT7hdoI9peIpPwV/+GNV5A5LyWPJSh6nzWs+yvqY3I
              </CipherData>
            - <ReferenceList>
                <DataReference URI="#EncryptedElement20"/>
              </ReferenceList>
            </EncryptedKey>
          </KeyInfo>
        - <CipherData>
            <CipherValue>dR2tmajCgBYjnxT6IIi3lLwpSfqs8E/L0Ng/EmPLHuVY8LWuPYRr0ZDLDRP
          </CipherData>
        </EncryptedData>
      </sel_reg>
    </selections>
  - <regprocess>
      <voter_reg_id_ori>SCSU_001</voter_reg_id_ori>
    </regprocess>
  - <valprocess>
      <val_data>4</val_data>
    </valprocess>
  - <admprocess>
      <election_data>5</election_data>
      <election_data>6</election_data>
    </admprocess>
  </eballot>
```


Figure 21. E-ballot with an encrypted voter ID.

Then the digital signature for the xml document is generated by using the singing process of the administration form as shown in Figure 22.

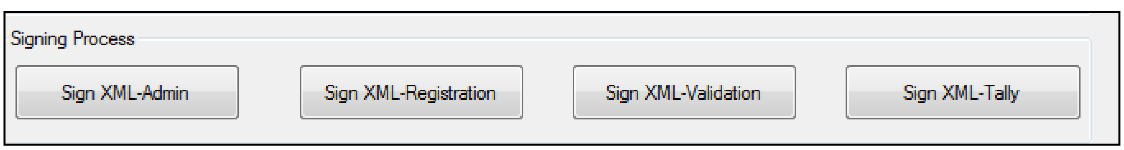


Figure 22. Signing process.

The generated digital signature for tally process is attached to the document as shown in Figure 23 (Signature code is inside the dashed line box).

```xml
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<eballot>
  - <selections id="t">
        <president>Ranolee Hettiarachchi</president>
        <secretory>John Kinght</secretory>
        <treasurer>Bob Steve</treasurer>
      + <sel_reg>
    </selections>
  - <regprocess>
        <voter_reg_id_ori>SCSU_001</voter_reg_id_ori>
    </regprocess>
  - <valprocess>
        <val_data>4</val_data>
    </valprocess>
  - <admprocess>
        <election_data>5</election_data>
        <election_data>6</election_data>
    </admprocess>
  - <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      - <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          - <Reference URI="#t">
              - <Transforms>
                    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue>uwBA7yNlKVRSe2vcxxkQrOwQJuY=</DigestValue>
            </Reference>
        </SignedInfo>
        <SignatureValue>6LmCfUwyp01x0fCUP+MFOdO7Rk6NNl7mKJpSkzmF2q1pMY+0uB+p4HAuLZPqbF92(
    </Signature>
</eballot>
```

Figure 23. E-ballot with a digital signature.

After placing the digital signature in the eballot-x.xml file, the data belonging to each process are encrypted with the combination of symmetric and asymmetric keys which belong to each process. The data of each process are encrypted with AES session keys (i.e., randomly generated symmetric keys) and the session keys are encrypted with unique RSA asymmetric keys (i.e., public key of the relevant processes). When sending the eballot-x.xml file to tally process, the main <eballot> element is encrypted by using the encryption keys of tally process. Other data in the electronic ballot are encrypted by using the encryption keys of relevant process to prevent unauthorized access from irrelevant parties (i.e., other processes or any other unauthorized external accesses).

The X-Ballot system generated separate pair of public and private keys for each process and store those in secure containers. Later the decryption processes can obtain their private keys from the secure container once they want to decrypt the data. The main administration form displayed the public keys of each process. Figure 24 shows a final encrypted file (i.e., Tally_eballot-x.xml) which is finally sent to the tally process. In this file, all sensitive information has been encrypted and replaced with the <EncryptedData> element. The X-Ballot system does the same operation for all available electronic ballots. These files are the inputs for the tally process to perform its tasks.


```
  <?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element" Id="EncryptedElement3">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
    - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        - <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
              <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
            - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                  <KeyName>rsaKeyTally</KeyName>
              </KeyInfo>
            - <CipherData>
                  <CipherValue>Uh8DAWMMVgpYmjpqZFGnp+Qf21qvZyMkfa9qZzz2hPX+hHGkB/QkrcZ4Y6uSehl
              </CipherData>
            - <ReferenceList>
                  <DataReference URI="#EncryptedElement3"/>
              </ReferenceList>
          </EncryptedKey>
      </KeyInfo>
    - <CipherData>
          <CipherValue>Et1SP8uxyEJPo1DLWGhkZ2avoZUZIGOMF0z69gnI1w8XOQzl+/PtGdPh64x9AK6N3D0Kj
      </CipherData>
  </EncryptedData>
```


Figure 24. Encrypted E-ballot-tally_eballot-x.xml.

When the tally process receives all the signed and encrypted electronic ballots, the tally process starts the decryption process. Figure 25 shows the decryption section of the tally process. The private key of the tally process is displayed only for illustration purposes in this prototype application.

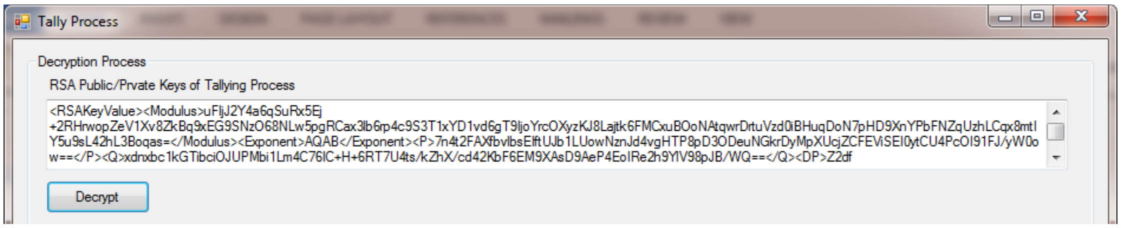


Figure 25. Decryption section of the tally process.

Figure 26 shows a decrypted e-ballot which belongs to the tally process. After successful decryption, the tally process can see the voter's candidate choices (i.e., data in the dashed line box) that are required to perform the main tasks of tally process. However, the tally process cannot see or use the other e-ballot data because its decryption process cannot decrypt data that are not belong to the tally process.

Similarly, other processes use their private keys to decrypt the contents and reveal the data that they are authorized to see or to perform their relevant operations. Once the decryption is over, the tally process can start the digital signature verification of the received e-ballots. Once the tally process administrator clicks on the "Verify Signature" button, the X-Ballot system automatically checks the signature of all e-ballots. Once the system finishes completing the verification, the system prompts a message as shown in Figure 27. Then the tally process administrator can click on "Check Integrity" button to view the results of the verification in the tally form as shown in Figure 28.

```
  <?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <eballot>
  - <selections id="t">
      <president>Ranolee Hettiarachchi</president>
      <secretory>John Kinght</secretory>
      <treasurer>Bob Steve</treasurer>
    + <sel_reg>
    </selections>
  - <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
    Type="http://www.w3.org/2001/04/xmlenc#Element" Id="EncryptedElement2">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
    - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
      - <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/:
        - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>rsaKeyReg</KeyName>
          </KeyInfo>
        - <CipherData>
            <CipherValue>o7b/w1/7W/nYYhxDLZtN4/zZDWJZXX4QKNvmJITiX3kT/CMd
          </CipherData>
        - <ReferenceList>
            <DataReference URI="#EncryptedElement2"/>
          </ReferenceList>
        </EncryptedKey>
      </KeyInfo>
    - <CipherData>
        <CipherValue>RcFNt5JJlE+xKCjdKv1UyRD2Dvf62AhrLzCFXNMq43eebrWcI+0MtDrJI
      </CipherData>
    </EncryptedData>
  + <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
    Type="http://www.w3.org/2001/04/xmlenc#Element" Id="EncryptedElement4">
  + <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
    Type="http://www.w3.org/2001/04/xmlenc#Element" Id="EncryptedElement5">
  - <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    - <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      - <Reference URI="#t">
        - <Transforms>
            <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-si
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <DigestValue>uwBA7yNlKVRSe2vcxxkQrOwQJuY=</DigestValue>
        </Reference>
      </SignedInfo>
      <SignatureValue>6LmCfUwyp01x0fCUP+MFOdO7Rk6NNl7mKJpSkzmF2q1pMY+0uB+p4
    </Signature>
</eballot>
```


Figure 26. Decrypted E-ballot of the tally process.

The X-Ballot system highlight the e-ballots in red which failed the signature verification as shown in Figure 27. Moreover, the system shows the number of e-ballots with failed verification. The failure of verification indicates that the data in those e-ballots have been tampered with. This mechanism helps the X-Ballot system to assure the integrity of e-ballot data any time after casting of electronic votes. In this way, each and every process can verify the integrity of their data without depending on another application.

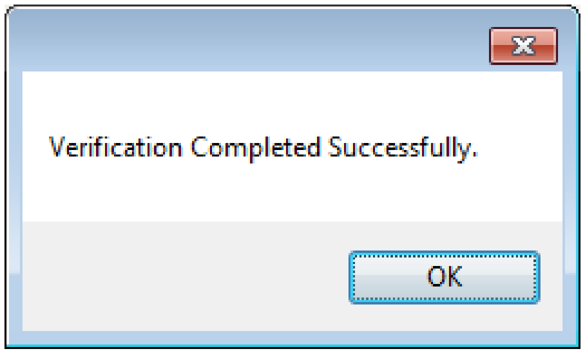


Figure 27. Digital signature verification completion message.

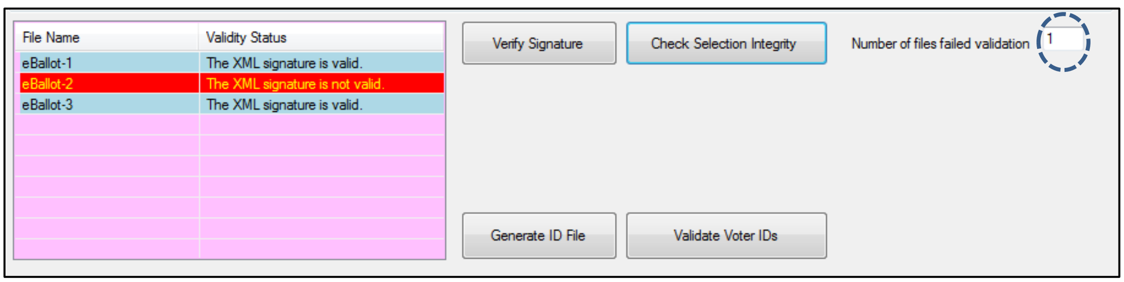


Figure 28. Digital signature validity status.

The X-Ballot system also proposes a mechanism to protect the privacy of voters. For example, if tally process needs to verify the voter's ID before starting the counting process, the tally process can request the administration process to validate voter's IDs. Neither the tally process nor administration process of X-Ballot system can see voter's ID and their vote together.

The tally process can generate an XML-based ID file (i.e., FileSendByTallyProcess.xml) which contains encrypted voter IDs which are received from the e-ballots as shown in Figure 29.

```
<?xml version="1.0"?>
<!-- This file send by tally proces to validate voter ids -->
- <voter-ids>
  + <sel_reg>
  + <sel_reg>
  - <sel_reg>
    - <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
      Type="http://www.w3.org/2001/04/xmlenc#Element" Id="EncryptedElement22">
        <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
      - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        - <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
          - <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
              <KeyName>pri_key</KeyName>
            </KeyInfo>
          - <CipherData>
              <CipherValue>WvmqC6VgxXJoLQI+zLNWjh4q1thiQO9sP+mACR1uFU4VJA3b96eRIP
            </CipherData>
          - <ReferenceList>
              <DataReference URI="#EncryptedElement22"/>
            </ReferenceList>
          </EncryptedKey>
        </KeyInfo>
      - <CipherData>
          <CipherValue>oyX4GaaSLPkknIezcsBtT2OnBpBtJQ0x0cTBkQedizGnix8eaaB8Nj4bDjP/6piYN
        </CipherData>
      </EncryptedData>
    </sel_reg>
</voter-ids>
```


Figure 29. FileSendByTallyProcess.xml.

The tally process doesn’t see voters’ ID in plain text. Therefore, the tally process sends a request to administration process to validate encrypted voter’s IDs in the “FileSendByTallyProcess.xml” file by pressing “Request Validate User IDs” button as shown in Figure 30.

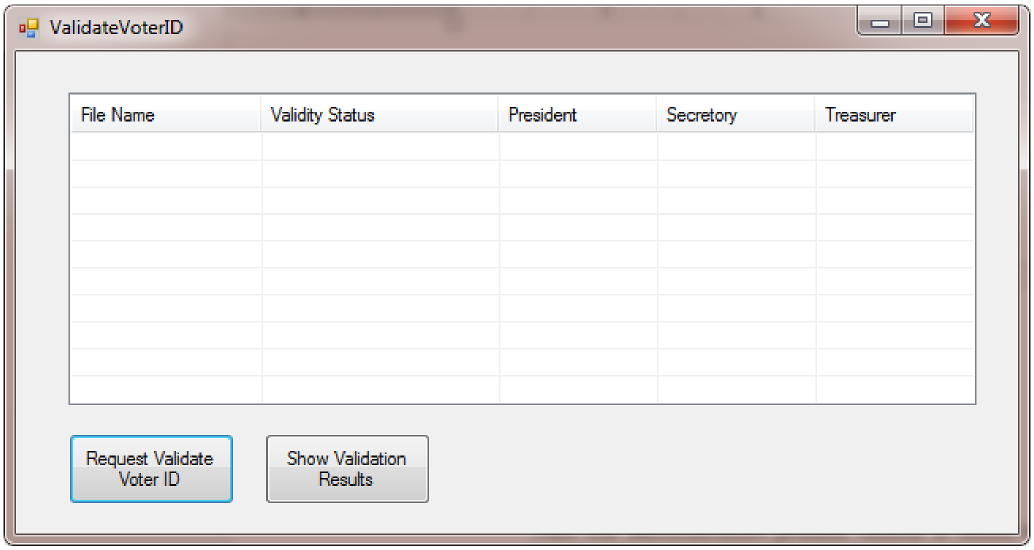


Figure 30. Voter ID validation form

Then the administration process receives a notification message about the request to validate voters IDs as shown in Figure 31.

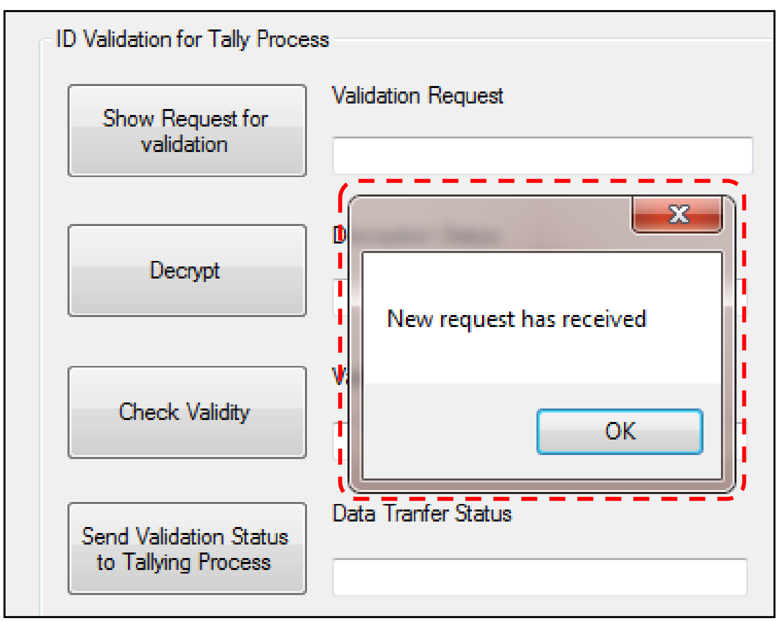


Figure 31. Administration process–voter ID validation request message

Once the administration process receives the request, the file is decrypted by the administration process and checks the validity of the voter IDs' sent by the tally process, as shown in Figure 32. The administration process compares the received IDs with not only the initial voter registration information file ("Voter_Registration.xml") but also with IDs recorded in the "CastedVotersIDs.xml" that contains only the IDs of voters who actually voted during the election time.

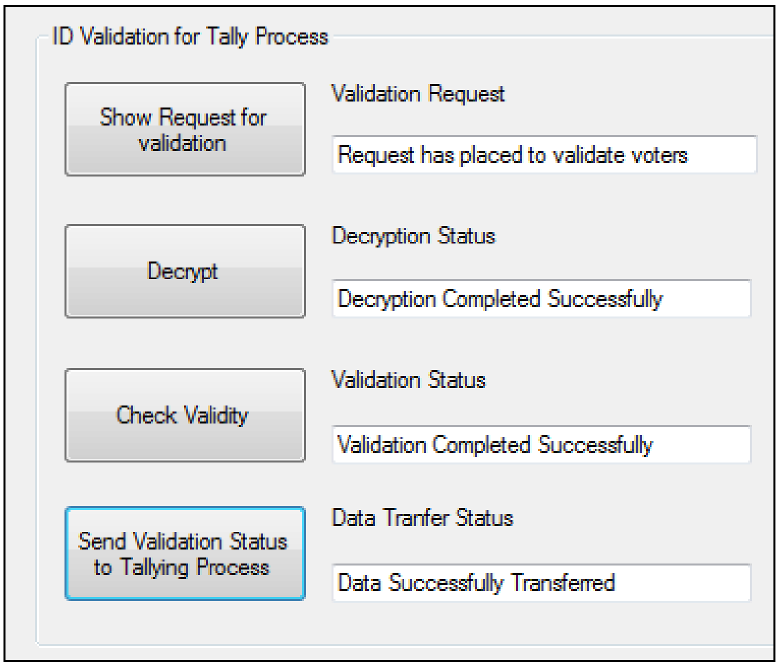


Figure 32. Voter ID validation.

Then the results (i.e., status of the IDs) are sent back to the tally process. The tally process can then tabulate the information as shown in Figure 33.

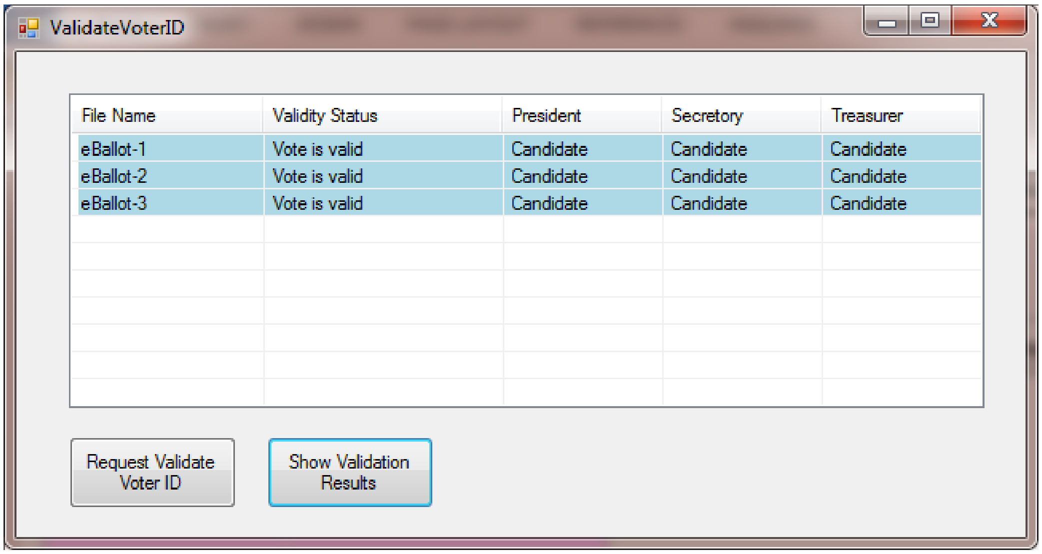


Figure 33. Voter ID validation-results.

After ID verification, the tally process proceeds to the counting sub-process and displays the final results as shown in Figure 34. The final election results can be viewed by pressing the "Display Results" button as shown in Figure 34. The full source code of the X-Ballot system is available in Appendix B.

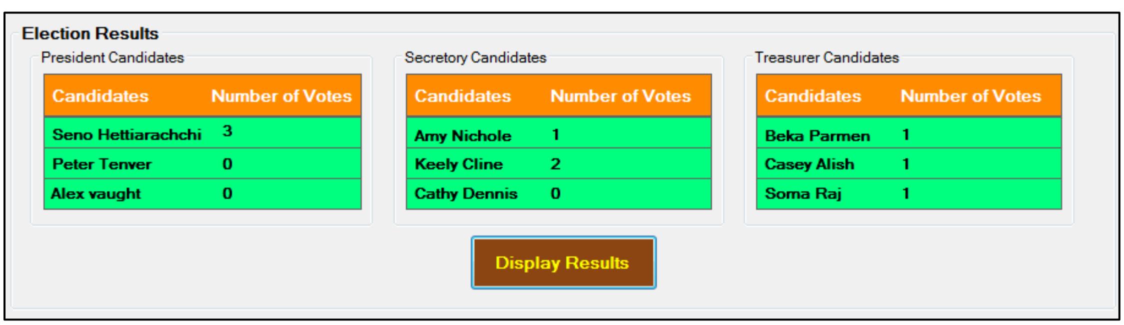


Figure 34. Election results.

Summary

This chapter covered the implementation of the proposed EVS which used the Microsoft .Net 2013 framework. The new EVS was developed as a prototype using

Visual C# language with the help of some in-built classes provided by the .Net framework. The chapter explained in detail the processes of the new voting system and its mechanism used to protect and monitor the integrity of the electronic ballots and the privacy of the voters. The next chapter explains how the new system was evaluated and compared with existing systems with respect to several criteria.

Chapter V: Discussion

Introduction

This chapter describes how the proposed X-Ballot system was evaluated using analytical hierarchy process (AHP). The new system was compared with two other existing systems with respect to four main criteria. The chapter explains in detail how the evaluation was conducted and also shows the data and calculations involved in the evaluation process.

Evaluation

Saaty (1990) introduced analytical hierarchy process which is a multi-criteria decision making method. The AHP process is commonly used for evaluating alternatives using multiple criteria. For example, an engineering field may use AHP when they want to make decision about the most appropriate technology among available technologies to perform a particular task. The AHP process enable us to use several criteria to evaluate alternatives. In this thesis, four main criteria were used to evaluate the proposed X-Ballot system and two other alternatives. The four criteria used in the AHP process are as follows:

i. Design and the main technologies used in the system (i.e., robustness/ efficiency/effectiveness/usefulness) (Criterion 1)

ii. Effectiveness in terms of integrity (Criterion 2)

iii. Cost effectiveness of the system (Criterion 3)

iv. Design for privacy protection (Criterion 4)

In this thesis, X-Ballot system and two other systems were used in the evaluation process. The three system used in this evaluation are as follows:

i. X-Ballot system (Alternative 1)

ii. EVS of North Dakota State University (Alternative 2)

iii. Verifiable EVS (Alternative 3)

The evaluation process was structured for all three systems as shown in Figure 35. Electronic voting system of North Dakota State University (NDSU) is used by the university community for online elections such as university student body election. Kaminski and Perry (2006) proposed verifiable electronic voting system (VEV) which is an open source electronic voting system designed to protect the secrecy of ballot data. Three volunteers who have several years of experience in the field of software engineering evaluated the three systems by means of AHP process. Evaluation results (i.e., priority vector values of three evaluators) were averaged to minimize the subjectivity of the process.

Figure 35. AHP hierarchy.

As the first step, the four criteria were compared using the pairwise comparison technique of the AHP process using the following scale as shown in Table 2 where 1 means same level of importance and 9 indicates extreme importance.

Table 2

Scales and Descriptions for AHP Pairwise Comparison

Scale	Degree of Preference
1	Equally Importance
2	Weak or slight
3	Moderate importance
4	Moderate plus
5	Strong importance
6	Strong plus
7	Very strong or demonstrated importance
8	Super strong
9	Extremely importance

Pairwise comparison matrix for the four criteria is shown in Table 3. The column total (CT) values of the matrix were used to normalize the comparison values in the matrix by dividing each value from the column total. Table 4 shows the priority vector (PV) values of the four criteria. These priority vector values were calculated by averaging the row sum of the matrix shown in Table 4. The three evaluators followed the same process individually and independently. After obtaining their PV values for the four criteria, their PV values for each criterion were averaged to get the final PV value for each criterion. Table 3 and 4 show the calculations of first evaluation done by the first evaluator.

Table 3

Comparison Matrix of Criteria

	C1	C2	C3	C4
C1	1	1/5	4	1/4
C2	5	1	8	3
C3	1/4	1/8	1	1/6
C4	4	1/3	6	1
CT	10 1/4	1 2/3	19	4 2/5

Table 4

Priority Vector Matrix of Criteria

	C1	C2	C3	C4	Total	PV
C1	0.098	0.121	0.211	0.057	0.485	0.121
C2	0.488	0.603	0.421	0.679	2.191	0.548
C3	0.024	0.075	0.053	0.038	0.190	0.048
C4	0.390	0.201	0.316	0.226	1.133	0.283

The resultant values need to be checked for consistency. The consistency check was performed using the equation 1 and 2. In order to accept the comparison values, Consistency Ratio (CR) value should be equal or less than 0.10 (10%).

$$Consistency\ Index\ (CI) = \frac{(\lambda max - n\ )}{(n-1)} \qquad \text{-------------------------------} \ (1)$$

λmax is the maximum eigenvalue of the matrix, and n is the order of the matrix.

$$Consistency\ Ratio\ (CR) = \frac{Consistency\ Index\ (CI)}{Random\ Index\ (RI)} \quad \text{-------------------------------} \ (2)$$

The random indexes (RI) for the matrices are shown in Table 5. RI values of 0.58 and 0.9 were used for the consistency check of the three alternatives and four criteria, respectively.

Table 5

Random Index (RI)

Random Index									
1	2	3	4	5	6	7	8	9	10
0.00	0.00	0.58	0.9	1.12	1.24	1.32	1.41	1.46	1.49

For the first criteria evaluation, the Consistency Ratio value was 0.078 (7.8%). This value is acceptable because it is less than 0.10 (10%). For each evaluation, CR value was calculated. Figure 36 shows the averaged Priority Vector values of the four criteria used in this thesis.

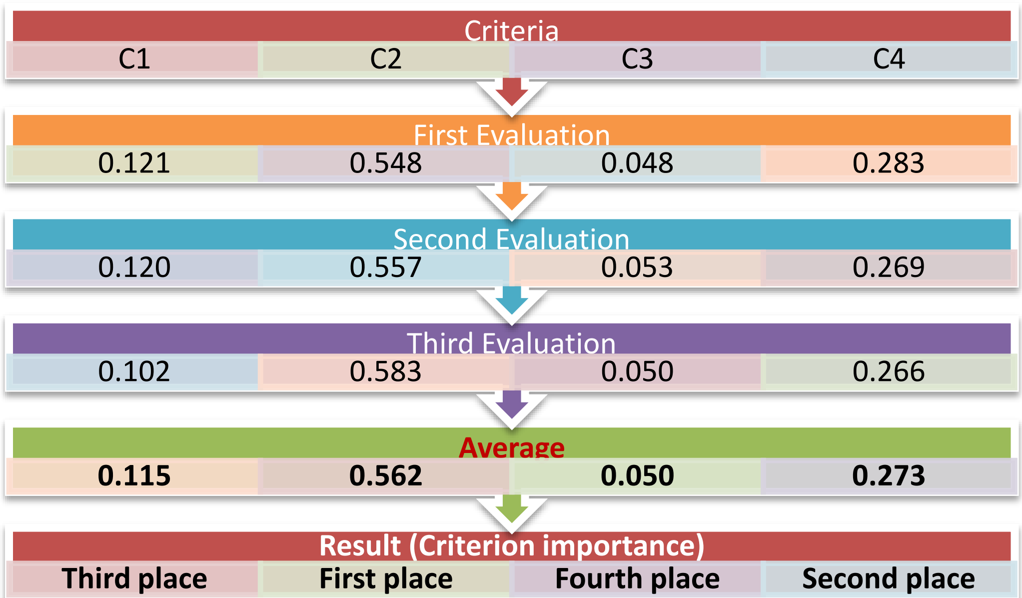


Figure 36. Averaged priority vectors and Rankings of criteria.

The aforementioned process was used to evaluate the three systems used in this thesis. Table 6 shows the pairwise comparison values of the three systems for the first criterion, technology & design. Similarly, Table 7, 8, and 9 show the comparison values of the three systems in terms of other three criteria[1]. Table 10, 11, 12, and 13 show the priority vectors (PV) for each system. The second and third evaluations followed the same process. Table 14 shows the averaged priority vectors for all three systems.

[1] The data shown in Tables from 6 to 13 belong only to the first evaluation.

Table 6

Design & Technology

	S1	S2	S3
S1	1	8	3
S2	1/8	1	1/4
S3	1/3	4	1
CT	1 1/2	13	4 1/4

Table 7

Safeguarding Integrity

	S1	S2	S3
S1	1	3	5
S2	1/3	1	2
S3	1/5	1/2	1
CT	1 1/2	4 1/2	8

Table 8

Cost Effectiveness

	S1	S2	S3
S1	1	4	1
S2	1/4	1	1/3
S3	1	3	1
CT	2 1/4	8	2 1/3

Table 9

Privacy Protection

	S1	S2	S3
S1	1	3	3
S2	1/3	1	2
S3	1/3	1/2	1
CT	1 2/3	4 1/2	6

Table 10

Priority Vector: Design & Technology

	S1	S2	S3	Total	PV
S1	0.686	0.615	0.706	2.007	0.669
S2	0.086	0.077	0.059	0.221	0.074
S3	0.229	0.308	0.235	0.772	0.257

Table 11

Priority Vector: Safeguarding Integrity

	S1	S2	S3	Total	PV
S1	0.652	0.667	0.625	1.944	0.648
S2	0.217	0.222	0.250	0.690	0.230
S3	0.130	0.111	0.125	0.367	0.122

Table 12

Priority Vector: Cost Effectiveness

	S1	S2	S3	Total	PV
S1	0.444	0.500	0.429	1.373	0.458
S2	0.111	0.125	0.143	0.379	0.126
S3	0.444	0.375	0.429	1.248	0.416

Table 13

Priority Vector: Privacy Protection

	S1	S2	S3	Total	PV
S1	0.600	0.667	0.500	1.767	0.589
S2	0.200	0.222	0.333	0.756	0.252
S3	0.200	0.111	0.167	0.478	0.159

Table 14

Averaged Priority Vector Values of the Three Systems

Criteria	Evaluations	System		
		System 1	System 2	System 3
Criteria 1	First Evaluation	0.669	0.074	0.257
	Second Evaluation	0.665	0.104	0.231
	Third Evaluation	0.688	0.078	0.234
	Average PV	**0.674**	**0.085**	**0.241**
Criteria 2	First Evaluation	0.648	0.230	0.122
	Second Evaluation	0.753	0.172	0.075
	Third Evaluation	0.723	0.174	0.103
	Average PV	**0.708**	**0.192**	**0.100**
Criteria 3	First Evaluation	0.458	0.126	0.416
	Second Evaluation	0.480	0.115	0.405
	Third Evaluation	0.429	0.143	0.429
	Average PV	**0.456**	**0.128**	**0.417**
Criteria 4	First Evaluation	0.589	0.252	0.159
	Second Evaluation	0.608	0.272	0.120
	Third Evaluation	0.707	0.201	0.092
	Average PV	**0.635**	**0.242**	**0.124**

In order to get the final results, all weight values of the three systems were multiplied by the ranking values of the criteria as follows[2].

$$\begin{bmatrix} 0.674 & 0.708 & 0.456 & 0.635 \\ 0.085 & 0.192 & 0.128 & 0.242 \\ 0.241 & 0.100 & 0.417 & 0.124 \end{bmatrix} \times \begin{bmatrix} 0.115 \\ 0.562 \\ 0.050 \\ 0.273 \end{bmatrix} \rightarrow \begin{bmatrix} 0.672 \\ 0.190 \\ 0.139 \end{bmatrix} \begin{array}{l} \Rightarrow \textbf{System 1} \text{ (X-Ballot)} \\ \Rightarrow \textbf{System 2} \text{ (NDSU Sys)} \\ \Rightarrow \textbf{System 3} \text{ (VEV)} \end{array}$$

The results of the evaluations shows that the proposed XML based system (i.e., X-Ballot system) obtained the highest value of 0.672. The final values for the NDSU e-voting system and the VEV are 0.19 and 0.13, respectively. Therefore, this results indicate that the X-Ballot system is an effective solution to secure electronic voting system in terms of integrity and privacy.

The evaluation process focused on the e-voting systems' capabilities of ensuring data integrity, privacy, the overall strength of the technologies used in the systems and the cost effectiveness of the implementation of the systems. The XML security techniques are the major technologies used in the design and the implementation of the proposed X-Ballot system. In particular, the strength of the XML-based encryption and digital signature techniques used in the X-Ballot system, supported to minimize integrity and privacy related concerns of electronic voting in a cost effective manner.

---

[2] Note that this is a matrix multiplication.

XML is the underlying technology for the design of the proposed X-Ballot system. All system generated files such as electronic ballots that are used by the voters to cast their vote and other required files are in the form of XML file format. The encrypted XML files received by the each process of the X-Ballot system ensure that unauthorized parties cannot access sensitive data in the files. The digital signature in each electronic ballot helps election officials to make sure the system consists of strong security measures to ensure the integrity (accuracy, consistency, and trustworthiness) of election data. Ensuring voters' privacy is another goal of this thesis. Therefore, the proposed X-Ballot system introduce a mechanism to verify e-ballot sensitive data such as IDs of voters without violating privacy of the voters.

In order to make sure all system generated XML files are well written and valid, all XML files in the X-Ballot system were validated (Markup Validation Service [APA], n.d.). In addition, the system generated keys for the encryption process are kept in secure containers.

Summary

This chapter described, how the X-Ballot system was evaluated in terms of four important properties of electronic voting systems. The new proposed system was compared with two other systems. All the steps involved in the evaluation process were explained in this chapter. The next chapter presents conclusion and possible future works of the new approach.

Chapter VI: Conclusions and Future Works

Introduction

This chapter presents a brief description of the thesis and summarizes the findings. The chapter also provides recommendations for future work related to the proposed new system.

Conclusion

Electronic voting is the electronic form of voting and it makes the voting process more accurate, efficient, and also helps to increase voter turnout. However, security is a major concern of EVS. Therefore, e-voting systems need to satisfy several important security requirements such as confidentiality, integrity, availability, and privacy. Protecting these essential security requirements is a very challenging task. Unauthorized parties can change sensitive data of e-voting systems for many different purposes through malicious attacks. Unauthorized modification of data in EVS, violate integrity requirement of the system. In order to prevent inappropriate use of electronic data, considering only the physical security is no longer adequate. Critical systems like EVS are highly vulnerable to various kinds of attacks. Therefore, these systems often need robust security measures and enhancements to protect important and critical data in their electronic ballots.

In this thesis, a new secure e-voting system was proposed to verify and safeguard the integrity and preserve the privacy of EVS in an efficient, less complicated, and also less expensive manner. The new EVS used advanced XML security standards such as XML encryption and XML Digital Signature, and also

Visual Studio.Net framework technologies. XML technologies used in this thesis significantly contributed to make the system very efficient. The proposed new approach was evaluated with two other existing EVS. The results showed that the new approach is very effective in verifying and defending integrity of electronic voting process as well as protecting privacy of voters. The proposed EVS is a better voting system for small and mid-size electoral populations. With the proposed approach, organizations can manage their voting process with confident. While the new approach reduces the effort needed to develop secure e-voting systems it also lowers the costs of developing e-voting systems for small and mid-size electoral populations.

Future Works

The proposed X-Ballot system designed to safeguard and detect integrity related issues of EVS and also to protect privacy. Therefore, the proposed system need to be improved to satisfy other security requirements of electronic voting such as confidentiality, nonrepudiation, and availability.

The proposed X-Ballot EVS is suitable for elections where the size of the election range from small to medium. The new system can be further improved to be used in large election processes with few modifications. A large election may have millions of voters. If an electronic election has a huge amount of votes, processing each and every e-ballot individually could be an inefficient approach. Therefore, at a certain point, a set of e-ballot data can be consolidated into a few separate files to reduce the burden and make the process more efficient and effective. This sort of improvements may also help reducing the network traffic and the high demand of computer resources such as computer memory, high performance central processing units, and high speed network connections.

References

Al-Hamdani, W. A. (2010). XML security in healthcare web systems. *Information Security Curriculum Development Conference*, 80-93.

Ansari, N., Sakarindr, P., Haghani, E., Zhang, C., Jain, A. K., & Shi, Y. Q. (2008). Evaluating electronic voting systems equipped with voter-verified paper records. *IEEE Security & Privacy*, *6*(3), 30-39. doi: 10.1109/MSP.2008.62

Bartel, M., Boyer, J., Fox, B., LaMacchia, B., & Simon, E. (2013, April 11). XML signature syntax and processing version 1.1. *World Wide Web Consortium (W3C)*. Retrieved from http://www.w3.org/TR/2013/REC-xmldsig-core1-20130411/.

Chia, T. (2012). Confidentiality, integrity, availability: The three components of the CIA Triad. Retrieved from http://security.blogoverflow.com/2012/08/confidentiality-integrity-availability-the-three-components-of-the-cia-triad/.

Ciampa, M. (2009). Basic cryptography. In *Security + guide to network security fundamentals* (pp. 365-389). Boston, MA: Course Technology, Cengage Learning.

Cranor, L. F., & Cytron, R. K. (1996*). Design and implementation of a practical security–conscious electronic polling system*. Washington University, Department of Computer Science.

Delaune, S., Kremer, S., & Ryan, M. (2006). Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of the 19th IEEE Workshop on Computer Security Foundations*, pp. 28-42.

Digital Signatures. (n.d.). *Resources and tools for IT professionals-TechNet*. Retrieved from http://technet.microsoft.com/en-us/library/cc962021.aspx.

Dournaee, B. (2002). Introduction to XML digital signatures part 1-2, XML signature examples. In *XML security* (pp. 107-220). New York, NY: McGraw-Hill.

Eastlake, D., & Niles, K. (2002). XML signature and authentication. In *Secure XML: The new syntax for signatures and encryption* (pp. 207-252). Boston, MA: Addison-Wesley.

Electronic voting systems. (2014). In *The ACE Electoral Knowledge Network*. Retrieved from http://aceproject.org/ace-en/topics/et/eth/eth02/eth02b/onePage.

Esteve, J. B., Goldsmith, B., & Turner, J. (2012). *International experience with e-voting*. Washington, DC: International Foundation for Electoral Systems.

Farivar, C. (2012). Internet-based and open source: How e-voting works around the globe. *Ars Technica*. Retrieved from http://arstechnica.com/features/2012/11/internet-based-and-open-source-how-e-voting-is-working-around-the-globe/.

Gerlach, J., & Gasser, U. (2009). Three case studies from Switzerland: E-voting. *Berkman Center Research Publication No.* 3, pp. 1-17.

How to: Sign XML documents with digital signatures. (n.d.). In *MSDN–the Microsoft Developer Network*. Retrieved from http://msdn.microsoft.com/en-us/library/ms229745.aspx.

How to: Verify the digital signatures of XML documents. (n.d.). In *MSDN–the Microsoft developer network*. Retrieved from http://msdn.microsoft.com/en-us/library/ms229950%28v=vs.100%29.aspx.

Ibrahim, S., Kamat, M., Salleh, M., & Aziz, S. R. A. (2003). Secure e-voting with blind signature. *Proceedings of 4th National Conference on Telecommunication Technology*, pp. 193-197.

Imamura, T., Clark, A., & Maruyama, H. (2002). A stream-based implementation of XML encryption. *Proceedings of the 2002 ACM Workshop on XML Security*, pp. 11-17.

Imamura, T., Dillaway, B., Simon, E., Yiu, K., & Nyström, M. (2013, April 11). XML Encryption Syntax and Processing Version 1.1. *World Wide Web Consortium (W3C)*. Retrieved from http://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/.

Kaminski, H., & Perry, M. (2006). Verifiable electronic voting system: An open source solution. *Computer Science Publications*, p. 11.

Konheim, A. (2007). The RSA cryptosystem. In *Computer security and cryptography* (pp. 376-382). Hoboken, NJ: John Wiley & Sons, Inc.

Krimmer, R. (2015). World map. Retrieved from https://www.e-voting.cc/en/it-elections/world-map/.

Kumar, V. K. N., & Srinivasan, B. (2013). A practical privacy preserving e-voting scheme with smart card using blind signature. *International Journal of*

Computer Network and Information Security, *5*(2), 42-50. doi: 10.5815/ijcnis.2013.02.06

Markup validation service. (n.d.). In *World Wide Web Consortium (W3C)*. Retrieved from https://validator.w3.org/.

Olaniyi, O. M., Arulogun, O. T., Omidiora, E. O., & Oludotun, A. (2013). Design of secure electronic voting system using multifactor authentication and cryptographic hash functions. *International Journal of Computer and Information Technology*, *2*(6), 1122-1130.

Pfleeger, C., & Pfleeger, S. (2007). Cryptography explained. In *Security in computing,* (pp. 748-755). Upper Saddle River, NJ: Prentice Hall Publications Office.

Rusinek, D., & Ksiezopolski, B. (2009). Voter non-repudiation oriented scheme for the medium scale e-voting protocol. *International Multiconference on Computer Science and Information Technology*, *4*, 325-330. doi: 10.1109/IMCSIT.2009.5352706

Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research, 48*(1), 9-28.

Santin, A. O., Costa, R. G., & Maziero, C. A. (2008). A three-ballot-based secure electronic voting system. *IEEE Security & Privacy*, *6*(3), 14-21. doi: 10.1109/MSP.2008.56

Sebe, F., Miret, J. M., Pujolis, J., & Puiggali, J. (2010). Simple and efficient hash-based verifiable mixing for remote electronic voting. *Journal Computer Communications, 33*(6), 667-675. doi: 10.1016/j.comcom.2009.11.013

Technology and the voting process. (2014). In *Elections Canada*. Retrieved from http://www.elections.ca/content.aspx?section=res&dir=rec/tech/tec&document=p7&lang=e.

Voter Validation Process FAQs. (2017). Retrieved from https://www.iccsafe.org/membership/voter-validation-process-faqs/.

Whitman, M., & Mattord, H. (2009). Cryptography. In *Principles of information security* (pp. 349-380). Boston, MA: Course Technology, Cengage Learning.

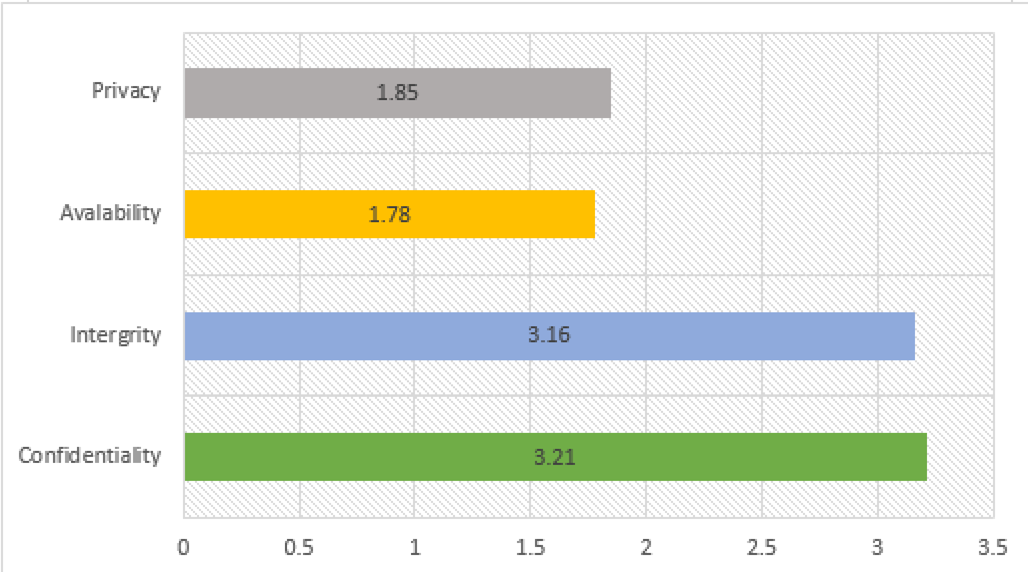
Wolf, P., Nackerdien, R., & Tuccinardi, D. (2011). Introducing electronic voting: Essential considerations. In *International Institute for Democracy and Electoral Assistance* (pp. 4-29). Stockholm, Sweden: International IDEA.

XDocument class overview (C#). (n.d.). In *MSDN–the Microsoft Developer Network.* Retrieved from https://msdn.microsoft.com/en-us/library/mt693057.aspx.

XML digital signature. (n.d.). In *MSDN–The Microsoft Developer Network*. Retrieved from http://msdn.microsoft.com/enus/library/windows/desktop/ms761350%28v=vs.85%29.aspx.

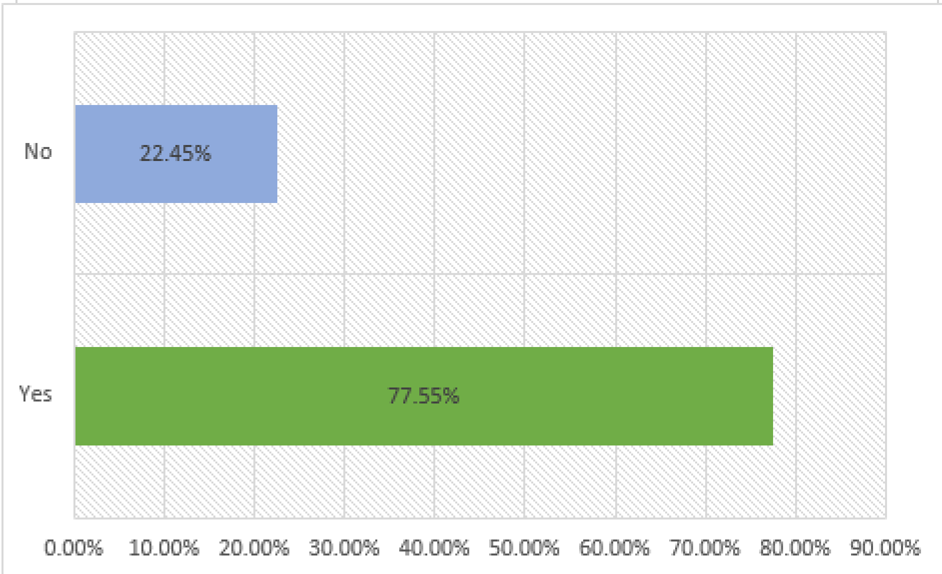
Appendix A: E-Voting Survey Questionnaire and Results

Q1: How do you rank the importance of the factors of e-voting system?



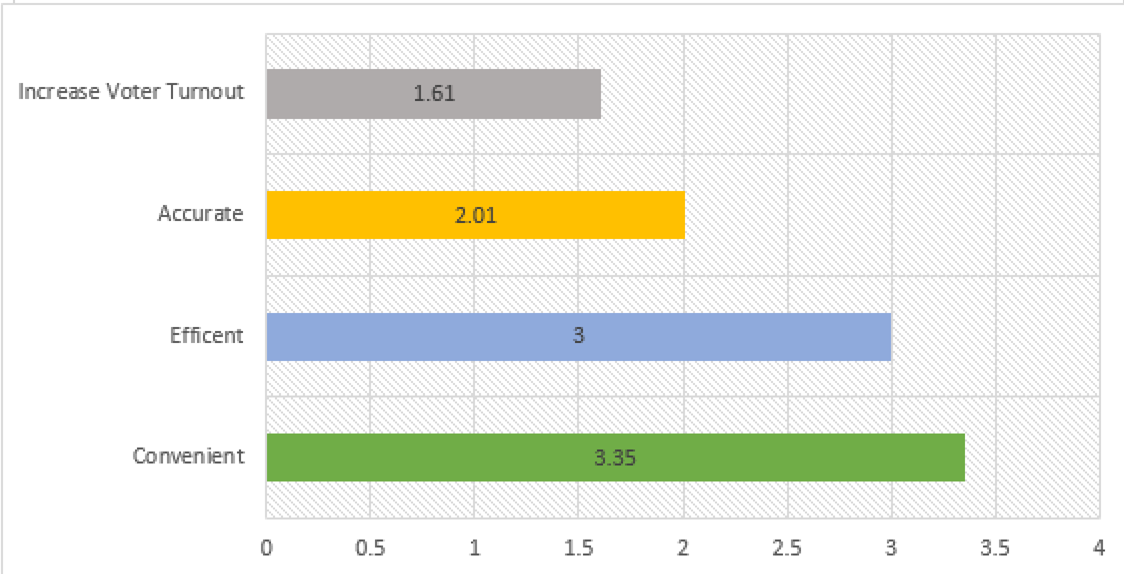
	1	2	3	4	Total	Score
Confidentiality	45	34	18	3	100	3.21
Intergrity	36	48	12	4	100	3.16
Avalability	3	4	61	32	100	1.78
Privacy	16	14	9	61	100	1.85

Q2: Do you think voting will eventually become completely electronic?



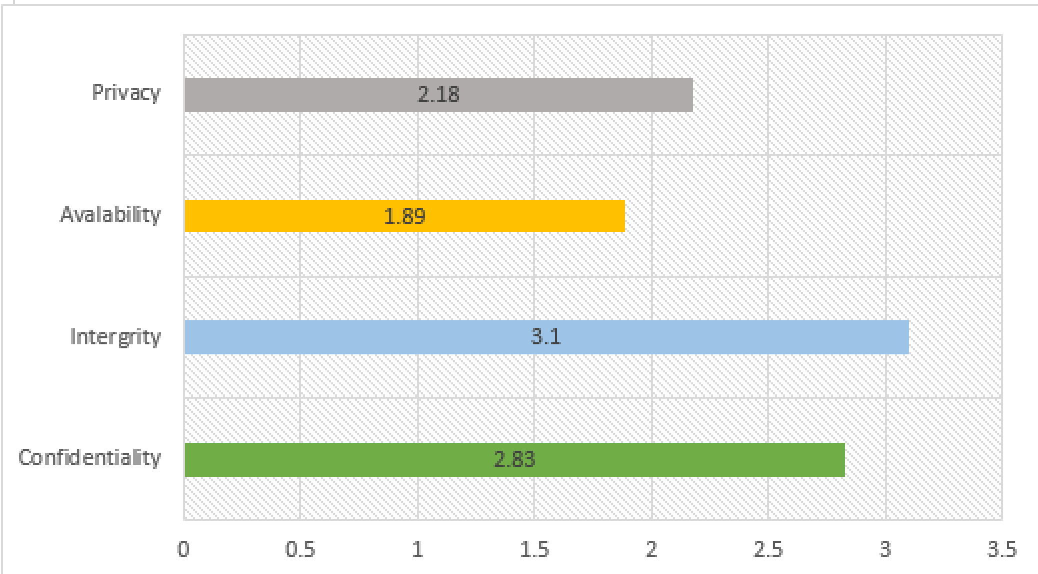
Answer Choices	Responses	
Yes	77.55%	76
No	22.45%	22
Total		**98**

Q3: How do you rank the following advantages of e-voting System?



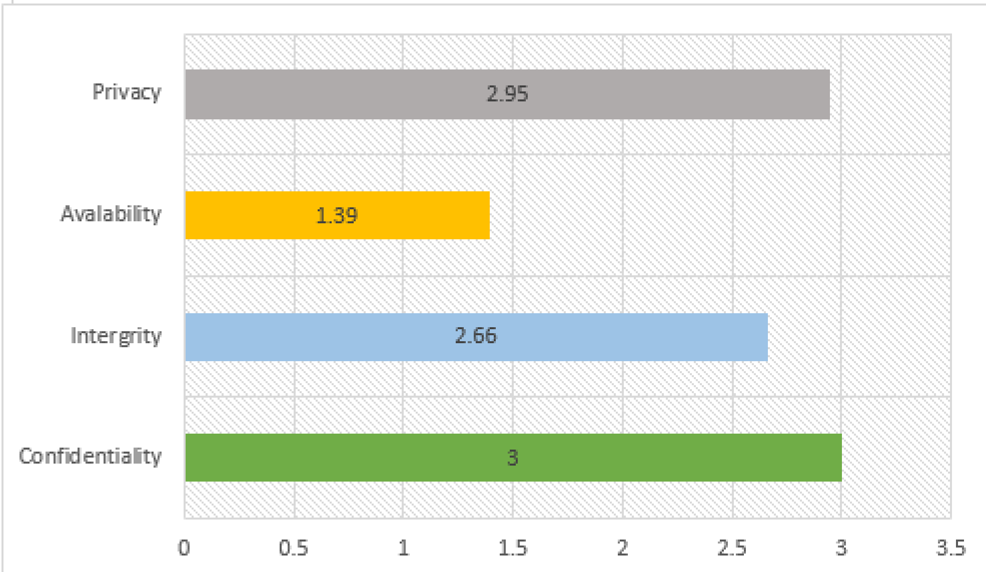
	1	2	3	4	Total	Score
Convenient	44	47	9	0	100	3.35
Efficent	34	36	29	1	100	3
Accurate	13	6	50	31	100	2.01
Increase Voter Turnout	9	11	12	68	100	1.61

Q4: How do you rank the following e-voting attributes according to potential threat level (Give one to highest)?



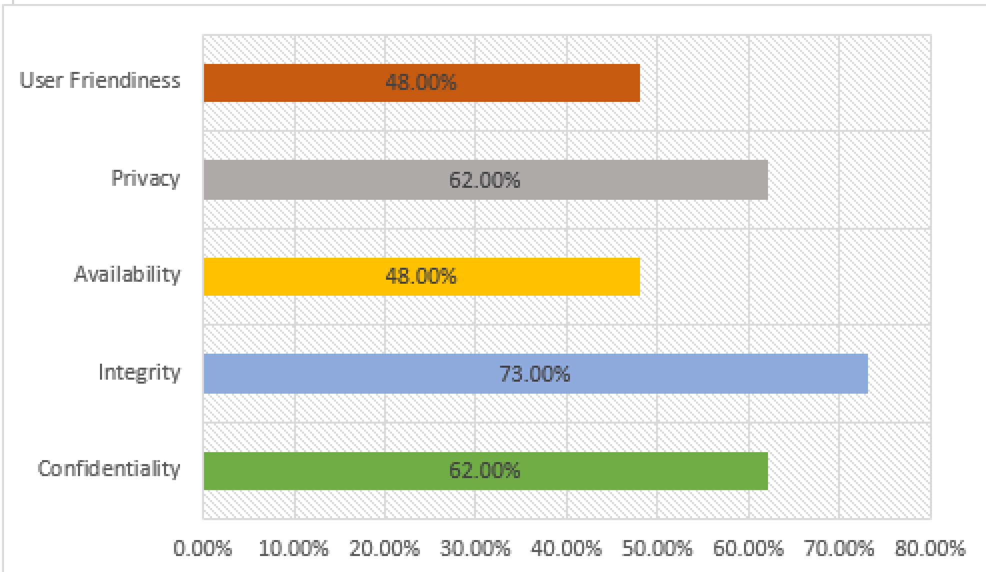
	1	2	3	4	Total	Score
Confidentiality	21	46	28	5	100	2.83
Intergrity	46	22	28	4	100	3.1
Avalability	16	6	29	49	100	1.89
Privacy	17	26	15	42	100	2.18

Q5: How do you rank the following e-voting attributes according to the level of harmfulness on voters if a malfunction take place in voting process (Give “one” to the most harmful attribute)?



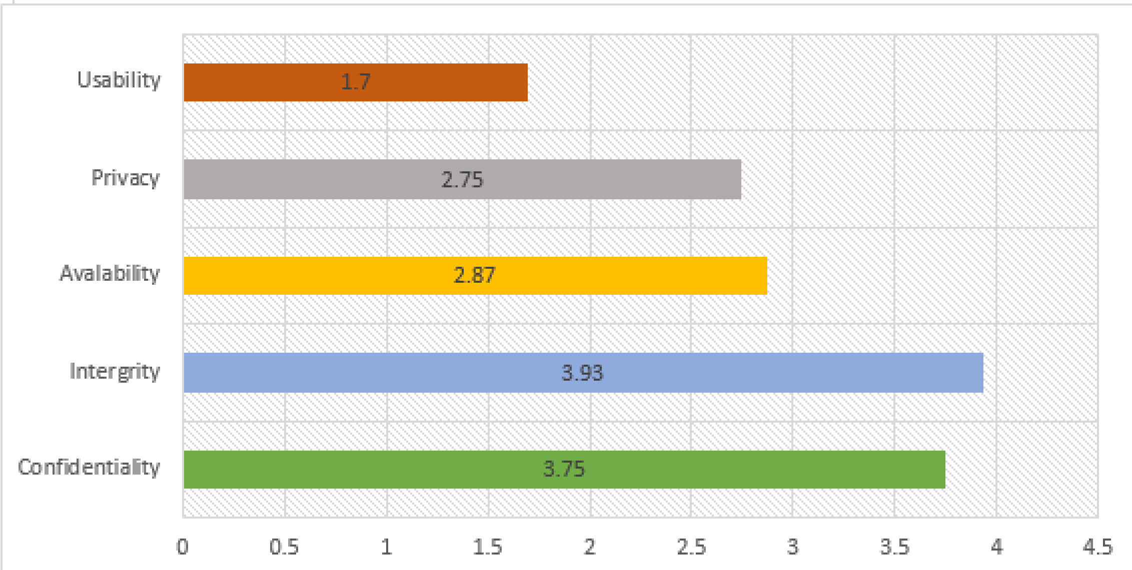
	1	2	3	4	Total	Score
Confidentiality	27	49	21	3	100	3
Intergrity	24	20	54	2	100	2.66
Avalability	6	3	15	76	100	1.39
Privacy	43	27	10	19	100	2.95

Q6: Select your top 3 e-voting attributes that you expect from an e-voting system as a voter.



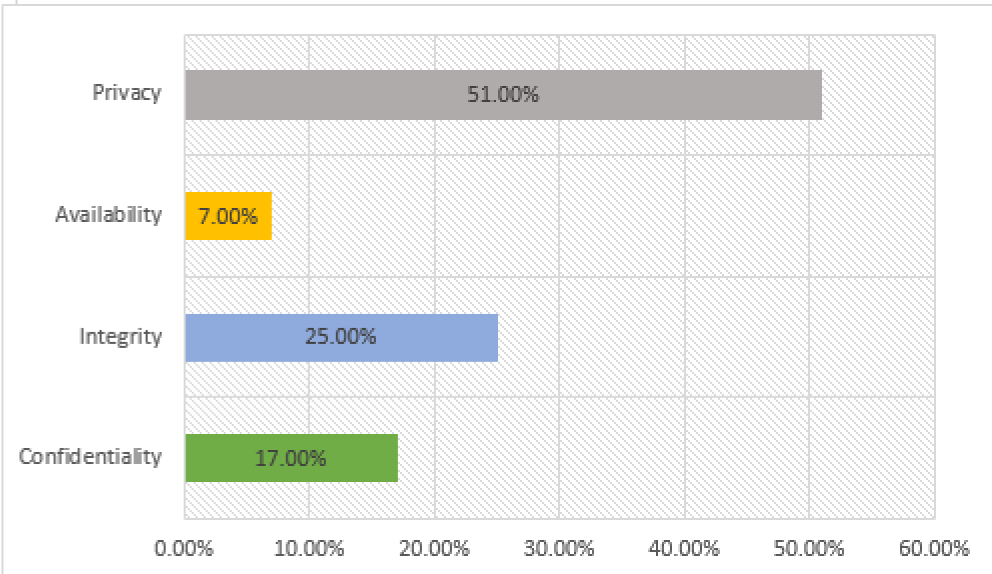
Answer Choices	**Responses**	
Confidentiality	62.00%	62
Integrity	73.00%	73
Availability	48.00%	48
Privacy	62.00%	62
User Friendiness	48.00%	48
Total Respondents: 100		

Q7: How do rank the following e-voting attributes according to the level of attention required by election authority (Ex: Election authority should pay more attention to your no “1” choice)?



	1	2	3	4	5	Total	Score
Confidentiality	22	42	26	9	1	100	3.75
Intergrity	40	25	26	6	3	100	3.93
Avalability	15	9	30	40	6	100	2.87
Privacy	14	19	12	38	17	100	2.75
Usability	9	5	6	7	73	100	1.7

Q8: In your opinion what is the e-voting attribute which has more social and/or ethical issues?



Answer Choices	Responses	
Confidentiality	17.00%	17
Integrity	25.00%	25
Availability	7.00%	7
Privacy	51.00%	51
Total		**100**

Appendix B: Source Code of X-Ballot System

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;


namespace ev_test
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainAdminForm());
        }
    }
}
```

MainAdminForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.Security.Cryptography;
using System.Security.Cryptography.Xml;

namespace ev_test
{
```

```
public partial class MainAdminForm : Form
{
    public string text2;
    //string[] selection_validity;
    string[] registered_voters;
    string[] send_by_tallyprocess;
    string[] validity_results_tp;
    string[] castedVotesID;

    public MainAdminForm()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Vote form1 = new Vote();
        form1.Show();
    }

    private void btnEncTally_Click(object sender, EventArgs e)
    {
        CspParameters cspParamsReg = new CspParameters();
        cspParamsReg.KeyContainerName = "XML_ENC_RSA_KEY_REG";

        CspParameters cspParamsRegAdm = new CspParameters();
        cspParamsRegAdm.KeyContainerName = "XML_ENC_RSA_KEY_REG_ADM";

        CspParameters cspParamsTally = new CspParameters();
        cspParamsTally.KeyContainerName = "XML_ENC_RSA_KEY_TALLY";

        CspParameters cspParamsValid = new CspParameters();
        cspParamsValid.KeyContainerName = "XML_ENC_RSA_KEY_VALID";

        RSACryptoServiceProvider rsaKeyReg      = new
RSACryptoServiceProvider(cspParamsReg);
        RSACryptoServiceProvider rsaKeyRegAdm   = new
RSACryptoServiceProvider(cspParamsRegAdm);
        RSACryptoServiceProvider rsaKeyTally    = new
RSACryptoServiceProvider(cspParamsTally);
```

```
        RSACryptoServiceProvider rsaKeyValid    = new
RSACryptoServiceProvider(cspParamsValid);


        this.txtRsaAdmPubKeyForTally.Text = rsaKeyRegAdm.ToXmlString(false);
        this.txtRsaValiPubKeyForTally.Text = rsaKeyValid.ToXmlString(false);
        this.txtRsaTallyPubKeyForTally.Text = rsaKeyTally.ToXmlString(false);
        this.txtRsaRegPubKeyForTally.Text = rsaKeyReg.ToXmlString(false);

        for (int i = 0; i < 3; i++)
        {
          XmlDocument xmlDoc = new XmlDocument();
          try
          {
            int ballotNumber = i + 1;
            string fn = "eballot-" + ballotNumber.ToString();

            string nfn = @"C:\votes\ballots\" + fn + ".xml";
            xmlDoc.PreserveWhitespace = true;
            xmlDoc.Load(nfn);
          }
          catch (Exception exp)
          {
            MessageBox.Show(exp.Message);
          }
          try
          {
            Encrypt(xmlDoc, "regprocess", "EncryptedElement2", rsaKeyReg, "rsaKeyReg");
            Encrypt(xmlDoc, "valprocess", "EncryptedElement4", rsaKeyValid,
"rsaKeyValid");
            Encrypt(xmlDoc, "admprocess", "EncryptedElement5", rsaKeyRegAdm,
"rsaKeyRegAdm");
            Encrypt(xmlDoc, "eballot", "EncryptedElement3", rsaKeyTally, "rsaKeyTally");
            int ballotNumber = i + 1;
            string fn2 = "Tally_eballot-" + ballotNumber.ToString();
            string nfn2 = @"C:\votes\" + fn2 + ".xml";
            xmlDoc.Save(nfn2);

          }
          catch (Exception exp)
          {
```

```
                MessageBox.Show(exp.Message);
            }
        }
                rsaKeyReg.Clear();
                rsaKeyTally.Clear();
                rsaKeyRegAdm.Clear();
                rsaKeyValid.Clear();
    }
        public static void Encrypt(XmlDocument Doc, string ElementToEncrypt, string
EncryptionElementID,  RSA Alg, string KeyName)
    {
        if (Doc == null)
            throw new ArgumentNullException("Doc");
        if (ElementToEncrypt == null)
            throw new ArgumentNullException("ElementToEncrypt");
        if (EncryptionElementID == null)
            throw new ArgumentNullException("EncryptionElementID");
        if (Alg == null)
            throw new ArgumentNullException("Alg");
        if (KeyName == null)

throw new ArgumentNullException("KeyName");
        XmlElement elementToEncrypt =
Doc.GetElementsByTagName(ElementToEncrypt)[0] as XmlElement;

        if (elementToEncrypt == null)
        {
            throw new XmlException("The specified element was not found");
        }
        RijndaelManaged sessionKey = null;
        try
        {
            sessionKey = new RijndaelManaged();
            sessionKey.KeySize = 256;

            EncryptedXml eXml = new EncryptedXml();

            byte[] encryptedElement = eXml.EncryptData(elementToEncrypt, sessionKey,
false);

            EncryptedData edElement = new EncryptedData();
```

```
                edElement.Type = EncryptedXml.XmlEncElementUrl;
                edElement.Id = EncryptionElementID;

                edElement.EncryptionMethod = new
EncryptionMethod(EncryptedXml.XmlEncAES256Url);
                EncryptedKey ek = new EncryptedKey();

                byte[] encryptedKey = EncryptedXml.EncryptKey(sessionKey.Key, Alg, false);

                ek.CipherData = new CipherData(encryptedKey);

                ek.EncryptionMethod = new EncryptionMethod(EncryptedXml.XmlEncRSA15Url);

                DataReference dRef = new DataReference();
                dRef.Uri = "#" + EncryptionElementID;
                ek.AddReference(dRef);

                edElement.KeyInfo.AddClause(new KeyInfoEncryptedKey(ek));
                KeyInfoName kin = new KeyInfoName();

                kin.Value = KeyName;

                ek.KeyInfo.AddClause(kin);
                edElement.CipherData.CipherValue = encryptedElement;
                EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);
            }
            catch (Exception exp)
            {
                throw exp;
            }
            finally
            {
                if (sessionKey != null)
                {
                    sessionKey.Clear();
                }
            }
        }

            private void btnTallyProcess_Click(object sender, EventArgs e)
            {
```

```
            TallyingProcess frm = new TallyingProcess();
            frm.Show();
        }

        private void btnEncReg_Click(object sender, EventArgs e)
        {
        CspParameters cspParamsReg = new CspParameters();
        cspParamsReg.KeyContainerName = "XML_ENC_RSA_KEY_REG";

        CspParameters cspParamsRegAdm = new CspParameters();
        cspParamsRegAdm.KeyContainerName = "XML_ENC_RSA_KEY_REG_ADM";

        CspParameters cspParamsTally = new CspParameters();
        cspParamsTally.KeyContainerName = "XML_ENC_RSA_KEY_TALLY";

        CspParameters cspParamsValid = new CspParameters();
        cspParamsValid.KeyContainerName = "XML_ENC_RSA_KEY_VALID";

        RSACryptoServiceProvider rsaKeyReg      = new
RSACryptoServiceProvider(cspParamsReg);
        RSACryptoServiceProvider rsaKeyRegAdm   = new
RSACryptoServiceProvider(cspParamsRegAdm);
        RSACryptoServiceProvider rsaKeyTally    = new
RSACryptoServiceProvider(cspParamsTally);
        RSACryptoServiceProvider rsaKeyValid    = new
RSACryptoServiceProvider(cspParamsValid);

        XmlDocument xmlDoc = new XmlDocument();
        try
        {
            xmlDoc.PreserveWhitespace = true;
            xmlDoc.Load("Vote_File.xml");
        }
        catch (Exception exp)
        {
            MessageBox.Show(exp.Message);
        }

        try
        {
            Encrypt(xmlDoc, "votes",       "EncryptedElement2", rsaKeyTally,   "rsaKeyTally");
```

```
                Encrypt(xmlDoc, "valprocess",   "EncryptedElement3", rsaKeyValid,
"rsaKeyValid");
                Encrypt(xmlDoc, "admprocess",   "EncryptedElement4", rsaKeyRegAdm,
"rsaKeyRegAdm");
                Encrypt(xmlDoc, "ballot",       "EncryptedElement5", rsaKeyReg,     "rsaKeyReg");

                xmlDoc.Save("Reg_Encrypted_Vote_File.xml");
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);
            }
            finally
            {
                rsaKeyReg.Clear();
                rsaKeyTally.Clear();
                rsaKeyRegAdm.Clear();
                rsaKeyValid.Clear();
            }
        }

        private void btnRegProcess_Click(object sender, EventArgs e)
        {
            RegProcess frmReg = new RegProcess();
            frmReg.Show();
        }

        private void btnAdminProcess_Click(object sender, EventArgs e)
        {
            AdmProcess frmAdm = new AdmProcess();
            frmAdm.Show();
        }

        private void btnValidationProcess_Click(object sender, EventArgs e)
        {
            ValiProcess frmVali = new ValiProcess();
            frmVali.Show();
        }

        private void btnEncAdm_Click(object sender, EventArgs e)
        {
```

```
            CspParameters cspParamsReg = new CspParameters();
            cspParamsReg.KeyContainerName = "XML_ENC_RSA_KEY_REG";

            CspParameters cspParamsRegAdm = new CspParameters();
            cspParamsRegAdm.KeyContainerName = "XML_ENC_RSA_KEY_REG_ADM";

            CspParameters cspParamsTally = new CspParameters();
            cspParamsTally.KeyContainerName = "XML_ENC_RSA_KEY_TALLY";

            CspParameters cspParamsValid = new CspParameters();
            cspParamsValid.KeyContainerName = "XML_ENC_RSA_KEY_VALID";

            RSACryptoServiceProvider rsaKeyReg      = new
RSACryptoServiceProvider(cspParamsReg);
            RSACryptoServiceProvider rsaKeyRegAdm   = new
RSACryptoServiceProvider(cspParamsRegAdm);
            RSACryptoServiceProvider rsaKeyTally    = new
RSACryptoServiceProvider(cspParamsTally);
            RSACryptoServiceProvider rsaKeyValid    = new
RSACryptoServiceProvider(cspParamsValid);

            XmlDocument xmlDoc = new XmlDocument();

            try
            {
               xmlDoc.PreserveWhitespace = true;
               xmlDoc.Load("Vote_File.xml");
            }
            catch (Exception exp)
            {
               MessageBox.Show(exp.Message);
            }
            try
            {
               Encrypt(xmlDoc, "votes",       "EncryptedElement2", rsaKeyTally,
"rsaKeyTally");
               Encrypt(xmlDoc, "regprocess",   "EncryptedElement3", rsaKeyReg,
"rsaKeyReg");
               Encrypt(xmlDoc, "valprocess",   "EncryptedElement4", rsaKeyValid,
"rsaKeyValid");
```

```
                Encrypt(xmlDoc, "ballot",      "EncryptedElement5", rsaKeyRegAdm,
"rsaKeyRegAdm");

                xmlDoc.Save("Admin_Encrypted_Vote_File.xml");
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);
            }
            finally
            {
                rsaKeyReg.Clear();
                rsaKeyTally.Clear();
                rsaKeyRegAdm.Clear();
                rsaKeyValid.Clear();
            }
        }
        private void btnEncVal_Click(object sender, EventArgs e)
        {
            CspParameters cspParamsReg = new CspParameters();
            cspParamsReg.KeyContainerName = "XML_ENC_RSA_KEY_REG";

            CspParameters cspParamsRegAdm = new CspParameters();
            cspParamsRegAdm.KeyContainerName = "XML_ENC_RSA_KEY_REG_ADM";

            CspParameters cspParamsTally = new CspParameters();
            cspParamsTally.KeyContainerName = "XML_ENC_RSA_KEY_TALLY";

            CspParameters cspParamsValid = new CspParameters();
            cspParamsValid.KeyContainerName = "XML_ENC_RSA_KEY_VALID";

            RSACryptoServiceProvider rsaKeyReg      = new
RSACryptoServiceProvider(cspParamsReg);
            RSACryptoServiceProvider rsaKeyRegAdm   = new
RSACryptoServiceProvider(cspParamsRegAdm);
            RSACryptoServiceProvider rsaKeyTally    = new
RSACryptoServiceProvider(cspParamsTally);
            RSACryptoServiceProvider rsaKeyValid    = new
RSACryptoServiceProvider(cspParamsValid);

            XmlDocument xmlDoc = new XmlDocument();
```

```
            try
            {
               xmlDoc.PreserveWhitespace = true;
               xmlDoc.Load(@"C:\votes\eballot-1.xml");
            }
            catch (Exception exp)
            {
               MessageBox.Show(exp.Message);
            }
            try
            {
               Encrypt(xmlDoc, "selections", "EncryptedElement7", rsaKeyTally, "rsaKeyTally");
               Encrypt(xmlDoc, "regprocess",   "EncryptedElement8",    rsaKeyReg,
"rsaKeyReg");
               Encrypt(xmlDoc, "admprocess",   "EncryptedElement9",    rsaKeyRegAdm,
"rsaKeyRegAdm");
               Encrypt(xmlDoc, "valprocess", "EncryptedElement10", rsaKeyValid,
"rsaKeyValid");

               xmlDoc.Save("Validate_Encrypted_e.xml");
            }
            catch (Exception exp)
         private void button1_Click_1(object sender, EventArgs e)
         {
            for (int i = 0; i < 3; i++)
            {
               try
               {
                  CspParameters cspParams = new CspParameters();
                  cspParams.KeyContainerName = "XML_DSIG_RSA_KEY";
                  RSACryptoServiceProvider rsaKey = new
RSACryptoServiceProvider(cspParams);
                  XmlDocument xmlDoc = new XmlDocument();

                  int ballotNumber = i + 1;
                  string fn = "eballot-" + ballotNumber.ToString();

                  string nfn = @"C:\votes\ballots\" + fn + ".xml";
                  xmlDoc.PreserveWhitespace = true;
                  xmlDoc.Load(nfn);
```

```
                SignXml(xmlDoc, rsaKey);
                xmlDoc.Save(nfn);
            }

            catch (Exception e1)
            {
                Console.WriteLine(e1.Message);
            }
        }
    }

    public static void SignXml(XmlDocument xmlDoc, RSA Key)
  {
    if (xmlDoc == null)
        throw new ArgumentException("xmlDoc");
    if (Key == null)
        throw new ArgumentException("Key");

    SignedXml signedXml = new SignedXml(xmlDoc);

    signedXml.SigningKey = Key;

    Reference reference = new Reference();
    reference.Uri = "#t";

    XmlDsigEnvelopedSignatureTransform env = new
XmlDsigEnvelopedSignatureTransform();
    reference.AddTransform(env);

    signedXml.AddReference(reference);

    signedXml.ComputeSignature();

    XmlElement xmlDigitalSignature = signedXml.GetXml();

    xmlDoc.DocumentElement.AppendChild(xmlDoc.ImportNode(xmlDigitalSignature,
true));
        }
        private void btnCheckValidity_Click(object sender, EventArgs e)
        {
            XmlDocument docTP = new XmlDocument();
```

```
XmlDocument docVR = new XmlDocument();
XmlDocument docCV = new XmlDocument();

docVR.Load("Voter_Registration.xml");
docTP.Load(@"C:\votes\FileSendByTallyProcess.xml");
docCV.Load("CastedVotersIDs.xml");
XmlNodeList elemListVR = docVR.GetElementsByTagName("voter_reg_id");
XmlNodeList elemListTP = docTP.GetElementsByTagName("voter_reg_id");
XmlNodeList elemListCV = docCV.GetElementsByTagName("voter_reg_id");
int size_elemListVR = elemListVR.Count;
int size_elemListTP = elemListTP.Count;
int size_elemListCV = elemListCV.Count;

registered_voters    = new string[size_elemListVR];
send_by_tallyprocess = new string[size_elemListTP];
castedVotesID        = new string[size_elemListCV];
validity_results_tp  = new string[size_elemListTP];

for (int i = 0; i < size_elemListVR; i++)
{
   registered_voters[i] = elemListVR[i].InnerText;
}

for (int i = 0; i < size_elemListCV; i++)
{
   castedVotesID[i] = elemListCV[i].InnerText;
}

for (int i = 0; i < size_elemListTP; i++)
{
   send_by_tallyprocess[i] = elemListTP[i].InnerText;
}

for (int i = 0; i < size_elemListTP; i++)
{
   int arrIndex = Array.IndexOf(registered_voters, send_by_tallyprocess[i]);

   if ((arrIndex > -1) && (send_by_tallyprocess[i] == castedVotesID[i]))
   {
      validity_results_tp[i] = "Vote is valid";
   }
```

```
            else
            {
                validity_results_tp[i] = "Vote is not valid";
            }
            Console.WriteLine(validity_results_tp[i]);
        }
        txtValidationStatus.Text = "Validation Completed Successfully";
    }

    private void button4_Click_1(object sender, EventArgs e)
    {
            CspParameters encPri = new CspParameters();
        encPri.KeyContainerName = "ENC_PRIVACY";

        RSACryptoServiceProvider pri_key = new RSACryptoServiceProvider(encPri);
        for (int i = 0; i < 3; i++)
        {
            XmlDocument xmlDoc = new XmlDocument();
            try
            {
                xmlDoc.PreserveWhitespace = true;
                xmlDoc.Load(@"C:\votes\FileSendByTallyProcess.xml");
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);
            }

            try
            {
                Decrypt(xmlDoc, pri_key, "pri_key");
                xmlDoc.Save(@"C:\votes\FileSendByTallyProcess.xml");

            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);
            }
        }
        pri_key.Clear();
```

```
            txtDecSta.Text = "Decryption Completed Successfully";
        }

    private void btnSendValStatus_TP_Click(object sender, EventArgs e)
        {
            ControlID2.TextData = validity_results_tp;
            txtDataTraSta.Text = "Data Successfully Transferred";
        }

        private void button5_Click(object sender, EventArgs e)
        {
            string text = ControlID.TextData;
            textBox2.Text = text;
        }

        public static void ShowMsg()
        {
            MessageBox.Show("New request has received");
        }

        public void btnEncForPrivacy_Click(object sender, EventArgs e)
        {
            CspParameters encPri = new CspParameters();
            encPri.KeyContainerName = "ENC_PRIVACY";
            RSACryptoServiceProvider pri_key = new RSACryptoServiceProvider(encPri);

            for (int i = 0; i < 3; i++)
            {
                XmlDocument xmlDoc = new XmlDocument();
                try
                {
                    int ballotNumber = i + 1;
                    string fnp = "eballot-" + ballotNumber.ToString();

                    string nfnp = @"C:\votes\ballots\" + fnp + ".xml";
                    xmlDoc.PreserveWhitespace = true;
                    xmlDoc.Load(nfnp);
                }
                catch (Exception exp)
                {
                    MessageBox.Show(exp.Message);
```

```
                }
                try
                {
                    int eleId = i + 20;
                    string eleId2 = "EncryptedElement" + eleId.ToString();

                    Encrypt(xmlDoc, "voter_reg_id", eleId2, pri_key, "pri_key");
                    int ballotNumber = i + 1;
                    string fnp = "eballot-" + ballotNumber.ToString();
                    string nfnp = @"C:\votes\ballots\" + fnp + ".xml";
                    xmlDoc.Save(nfnp);
                }
                catch (Exception exp2)
                {
                    MessageBox.Show(exp2.Message);
                }
            }
                    pri_key.Clear();
        }

        public static void Encrypt_privacy(XmlDocument Doc, string ElementName,
SymmetricAlgorithm Key)
        {
            if (Doc == null)
                throw new ArgumentNullException("Doc");
            if (ElementName == null)
                throw new ArgumentNullException("ElementToEncrypt");
            if (Key == null)
                throw new ArgumentNullException("Alg");
            XmlElement elementToEncrypt =
Doc.GetElementsByTagName(ElementName)[0] as XmlElement;

            if (elementToEncrypt == null)
            {
                throw new XmlException("The specified element was not found");
            }
            EncryptedXml eXml = new EncryptedXml();
            byte[] encryptedElement = eXml.EncryptData(elementToEncrypt, Key, false);

            EncryptedData edElement = new EncryptedData();
            edElement.Type = EncryptedXml.XmlEncElementUrl;
```

```
            string encryptionMethod = null;

            if (Key is TripleDES)
            {
                encryptionMethod = EncryptedXml.XmlEncTripleDESUrl;
            }
            else if (Key is DES)
            {
                encryptionMethod = EncryptedXml.XmlEncDESUrl;
            }
            if (Key is Rijndael)
            {
                switch (Key.KeySize)
                {
                    case 128:
                        encryptionMethod = EncryptedXml.XmlEncAES128Url;
                        break;
                    case 192:
                        encryptionMethod = EncryptedXml.XmlEncAES192Url;
                        break;
                    case 256:
                        encryptionMethod = EncryptedXml.XmlEncAES256Url;
                        break;
                }
            }
            else
            {
                throw new CryptographicException("The specified algorithm is not supported
for XML Encryption.");
            }

            edElement.EncryptionMethod = new EncryptionMethod(encryptionMethod);
            edElement.CipherData.CipherValue = encryptedElement;

            EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);
        }

        public static void Decrypt(XmlDocument Doc, RSA Alg, string KeyName)
        {
            if (Doc == null)
```

```
                throw new ArgumentNullException("Doc");
            if (Alg == null)
                throw new ArgumentNullException("Alg");
            if (KeyName == null)
                throw new ArgumentNullException("KeyName");

            EncryptedXml exml = new EncryptedXml(Doc);

            exml.AddKeyNameMapping(KeyName, Alg);

            exml.DecryptDocument();
        }
    }

    public static class ControlID2
    {
        public static string [] TextData { get; set; }
    }
}
```

Vote.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml.Linq;
using System.Xml;
using System.Xml.Serialization;
using System.Security.Cryptography;
using System.Security.Cryptography.Xml;
using System.IO;

namespace ev_test
{
    public partial class Vote : Form
    {
        public Vote()
```

```
    {
        InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        for (int i = 1; ; i++)
        {
            string fn = "eballot-" + i.ToString();
            string nfn = @"C:\votes\ballots\" + fn + ".xml";
            if (!File.Exists(nfn))
            {
                XDocument xdoc = new XDocument(
                    new XDeclaration("1.0", "utf-8", "yes"),
                    new XElement("eballot",
                        new XElement("selections", new XAttribute("id", "t"),
                            new XElement("president", lblDispalyPre.Text),
                            new XElement("secretory", lblDispalyVicePre.Text),
                            new XElement("treasurer", lblDispalyTreasurer.Text),
                            new XElement("sel_reg",
                                new XElement("voter_reg_id", txtVoterTallyId.Text))),
                        new XElement("regprocess",
                            new XElement("voter_reg_id_ori", txtVoterID.Text)),
                        new XElement("valprocess",
                             new XElement("val_data", "4")),
                        new XElement("admprocess",
                             new XElement("election_data", "5"),
                             new XElement("election_data", "6"))
                                    ));
                xdoc.Save(nfn);

                txtVoterID.Text = "";
                txtVoterTallyId.Text = "";
                lblDispalyPre.Text = "Your selection is displayed here";
                lblDispalyVicePre.Text = "Your selection is displayed here";
                lblDispalyTreasurer.Text = "Your selection is displayed here";

                break;
            }
            else
            {
            }
```

```
        }
    }
    private void btnPreCh1_Click(object sender, EventArgs e)
    {
        lblDispalyPre.Text = btnPreCh1.Text;
    }

    private void btnPreCh2_Click(object sender, EventArgs e)
    {
        lblDispalyPre.Text = btnPreCh2.Text;
    }

    private void btnVicePreCh1_Click(object sender, EventArgs e)
    {
        lblDispalyVicePre.Text = btnVicePreCh1.Text;
    }

    private void btnVicePreCh2_Click(object sender, EventArgs e)
    {
        lblDispalyVicePre.Text = btnVicePreCh2.Text;
    }

    private void btnVicePreCh3_Click(object sender, EventArgs e)
    {
        lblDispalyVicePre.Text = btnVicePreCh3.Text;
    }

    private void btnPreCh3_Click(object sender, EventArgs e)
    {
        lblDispalyPre.Text = btnPreCh3.Text;
    }

    private void btnTreCh1_Click(object sender, EventArgs e)
    {
        lblDispalyTreasurer.Text = btnTreCh1.Text;
    }

    private void btnTreCh2_Click(object sender, EventArgs e)
    {
        lblDispalyTreasurer.Text = btnTreCh2.Text;
    }
```

```
        private void btnTreCh3_Click(object sender, EventArgs e)
        {
            lblDispalyTreasurer.Text = btnTreCh3.Text;
        }
    }
}
```

TallyingProcess.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.Security.Cryptography;
using System.Security.Cryptography.Xml;

namespace ev_test
{
    public partial class TallyingProcess : Form
    {
        string[] selection_validity;
        int count_invalid_tallyprocess = 0;

        string[] validity_results_tp = { "No prior request to validate" };

        public TallyingProcess()
        {
            InitializeComponent();
        }
        private void btnDecTally_Click(object sender, EventArgs e)
        {
            CspParameters cspParamsTally = new CspParameters();
            cspParamsTally.KeyContainerName = "XML_ENC_RSA_KEY_TALLY";

            RSACryptoServiceProvider rsaKeyTally = new
RSACryptoServiceProvider(cspParamsTally);
```

```
        this.txtTallyPriKey.Text = rsaKeyTally.ToXmlString(true);

        for (int i = 0; i < 3; i++)
            {
            XmlDocument xmlDoc = new XmlDocument();
            try
            {
            int ballotNumber = i + 1;
            string fn3 = "Tally_eballot-" + ballotNumber.ToString();

                string nfn3 = @"C:\votes\" + fn3 + ".xml";

            xmlDoc.PreserveWhitespace = true;
            xmlDoc.Load(nfn3);
            }
            catch (Exception exp)
            {
            MessageBox.Show(exp.Message);
            }
            try
            {
            Decrypt(xmlDoc, rsaKeyTally, "rsaKeyTally");

            int ballotNumber = i + 1;
            string fn4 = "Tally_eballot-" + ballotNumber.ToString();

            string nfn4 = @"C:\votes\" + fn4 + ".xml";
            xmlDoc.Save(nfn4);
            }
            catch (Exception exp)
            {
            MessageBox.Show(exp.Message);
            }
            }
            rsaKeyTally.Clear();
    }

    public static void Decrypt(XmlDocument Doc, RSA Alg, string KeyName)
    {
        if (Doc == null)
            throw new ArgumentNullException("Doc");
```

```
        if (Alg == null)
            throw new ArgumentNullException("Alg");
        if (KeyName == null)
            throw new ArgumentNullException("KeyName");

        EncryptedXml exml = new EncryptedXml(Doc);

        exml.AddKeyNameMapping(KeyName, Alg);

        exml.DecryptDocument();
    }

    private void btnShowResults_Click(object sender, EventArgs e)
    {

        int VoteRH = 0;
        int VoteSA = 0;
        int VoteSK = 0;
        int VoteJK = 0;
        int VoteIA = 0;
        int VoteGS = 0;
        int VoteBob = 0;
        int VoteAle = 0;
        int VoteTar = 0;

        for (int j = 0; j < 3; j++)
        {
            XmlDocument xmlDoc = new XmlDocument();


            int ballotNumber = j + 1;
            string fnt = "Tally_eballot-" + ballotNumber.ToString();
            string nfnt = @"C:\votes\" + fnt + ".xml";
            xmlDoc.Load(nfnt);

            XmlElement eballot = xmlDoc.DocumentElement;
            XmlNodeList elemList = eballot.GetElementsByTagName("president");

                if (elemList[0].InnerXml == "Seno Hettiarachchi")
                    VoteRH = VoteRH + 1;
                if (elemList[0].InnerXml == "Peter Tenver")
```

```
                VoteSA = VoteSA + 1;
            if (elemList[0].InnerXml == "Alex Vaught")
                VoteSK = VoteSK + 1;

        lblRH.Text = VoteRH.ToString();
        lblSA.Text = VoteSA.ToString();
        lblSK.Text = VoteSK.ToString();

        XmlNodeList elemList2 = eballot.GetElementsByTagName("secretory");

            if (elemList2[0].InnerXml == "Amy Nicole")
                VoteJK = VoteJK + 1;
            if (elemList2[0].InnerXml == "Keely Cline")
                VoteIA = VoteIA + 1;
            if (elemList2[0].InnerXml == "Cathy Dennis")
                VoteGS = VoteGS + 1;

        lblJK.Text = VoteJK.ToString();
        lblIA.Text = VoteIA.ToString();
        lblGS.Text = VoteGS.ToString();

        XmlNodeList elemList3 = eballot.GetElementsByTagName("treasurer");

        if (elemList3[0].InnerXml == "Beka Parman")
            VoteBob = VoteBob + 1;
        if (elemList3[0].InnerXml == "Casey Alish")
            VoteAle = VoteAle + 1;
        if (elemList3[0].InnerXml == "Soma Raj")
            VoteTar = VoteTar + 1;

        lblBob.Text = VoteBob.ToString();
        lblAle.Text = VoteAle.ToString();
        lblTar.Text = VoteTar.ToString();
    }
}

private void btnValidateID_Click(object sender, EventArgs e)
{
    ValidateVoterID frmVVID = new ValidateVoterID();
    frmVVID.Show();
}
```

```
private void btn_VerifySig_Click(object sender, EventArgs e)
{
    System.IO.DirectoryInfo dir = new System.IO.DirectoryInfo(@"C:\votes\ballots\");
    int noeb = dir.GetFiles().Length;

    selection_validity = new string[noeb];
    for (int i = 0; i < noeb; i++)
    {
        try
        {
            CspParameters cspParams = new CspParameters();
            cspParams.KeyContainerName = "XML_DSIG_RSA_KEY";
            RSACryptoServiceProvider rsaKey = new
RSACryptoServiceProvider(cspParams);

            XmlDocument xmlDoc = new XmlDocument();
            int ballotNumber = i + 1;
            string fn = "Tally_eballot-" + ballotNumber.ToString();
            string nfn = @"C:\votes\" + fn + ".xml";
            xmlDoc.PreserveWhitespace = true;
            xmlDoc.Load(nfn);

            bool result = VerifyXml(xmlDoc, rsaKey);
            if (result)
            {
             selection_validity[i] = "The XML signature is valid.";
            }
            else
            {
                selection_validity[i] = "The XML signature is not valid.";
                count_invalid_tallyprocess = count_invalid_tallyprocess + 1;
            }
        }
        catch (Exception e2)
        {
            Console.WriteLine(e2.Message);
        }
    }
    MessageBox.Show("Verification Completed Successfully.");
}
```

```
        public static Boolean VerifyXml(XmlDocument Doc, RSA Key)
        {
            if (Doc == null)
                throw new ArgumentException("Doc");
            if (Key == null)
                throw new ArgumentException("Key");
            SignedXml signedXml = new SignedXml(Doc);
            XmlNodeList nodeList = Doc.GetElementsByTagName("Signature");

            if (nodeList.Count <= 0)
            {
                throw new CryptographicException("Verification failed: No Signature was found in
the document.");
            }
            if (nodeList.Count >= 2)
            {
                throw new CryptographicException("Verification failed: More than one signature
was found for the document.");
            }
            signedXml.LoadXml((XmlElement)nodeList[0]);
            return signedXml.CheckSignature(Key);
        }

        private void btnCSI_Tally_Click(object sender, EventArgs e)
        {
            listViewSelections.Clear();

            listViewSelections.View = View.Details;
            listViewSelections.GridLines = true;
            listViewSelections.FullRowSelect = true;

            listViewSelections.Columns.Add("File Name", 150);
            listViewSelections.Columns.Add("Validity Status", 200);

            for (int i = 0; i < selection_validity.Length; i++)
            {
                string[] arr = new string[selection_validity.Length];
                ListViewItem itm;

                int ballotNumber = i + 1;
```

```
                arr[0] = "eBallot-" + ballotNumber;
                arr[1] = selection_validity[i];
                itm = new ListViewItem(arr);
                if (arr[1] == "The XML signature is not valid.")
                {
                    itm.BackColor = Color.Red;
                    itm.ForeColor = Color.Yellow;
                }
                else
                {
                    itm.BackColor = Color.LightBlue;
                }
                listViewSelections.Items.Add(itm);
            }
            txb_CountInvalidTP.Text = count_invalid_tallyprocess.ToString();
        }
        private void btnGenIdFile_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < 3; i++)
            {
                XmlDocument doc5 = new XmlDocument();
                doc5.Load(@"C:\votes\FileSendByTallyProcess.xml");

                int ballotNumber = i + 1;
                string fnDec = "Tally_eballot-" + ballotNumber.ToString();

                string nfnDec = @"C:\votes\" + fnDec + ".xml";

                XmlDocument doc6 = new XmlDocument();
                doc6.Load(nfnDec);

                XmlNode newVoteID =
doc5.ImportNode(doc6.SelectSingleNode("/eballot/selections/sel_reg"), true);
                doc5.DocumentElement.AppendChild(newVoteID);

                doc5.Save(@"C:\votes\FileSendByTallyProcess.xml");
                doc6.Save(nfnDec);
            }
        }
    }
}
```

ValidateVoterID.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ev_test
{
    public partial class ValidateVoterID : Form
    {
        public ValidateVoterID()
        {
            InitializeComponent();
        }

        private void btnShowValResults_Click(object sender, EventArgs e)
        {
            string[] validity_results_tp2 = ControlID2.TextData;

            try
            {
            listViewValResult.Clear();

            listViewValResult.View = View.Details;
            listViewValResult.GridLines = true;
            listViewValResult.FullRowSelect = true;

            listViewValResult.Columns.Add("File Name", 120);
            listViewValResult.Columns.Add("Validity Status", 150);
            listViewValResult.Columns.Add("President", 100);
            listViewValResult.Columns.Add("Secretory", 100);
            listViewValResult.Columns.Add("Treasurer", 100);

            for (int i = 0; i < validity_results_tp2.Length; i++)
            {
                string[] arr = new string[5];
```

```
                ListViewItem itm;

                int ballotNumber = i + 1;
                arr[0] = "eBallot-" + ballotNumber;
                arr[1] = validity_results_tp2[i];
                arr[2] = "Candidate";
                arr[3] = "Candidate";
                arr[4] = "Candidate";
                itm = new ListViewItem(arr);
                if (arr[1] == "Vote is valid")
                {
                    itm.BackColor = Color.LightBlue;
                }
                else
                {
                    itm.BackColor = Color.Red;
                    itm.ForeColor = Color.Yellow;
                }
                listViewValResult.Items.Add(itm);
            }
        }
                catch (Exception exp)
                {
                    MessageBox.Show(exp.Message);
                }
        }

        private void btnRequestValidate_Click(object sender, EventArgs e)
        {
            ControlID.TextData = "Request has placed to validate voters";
            MainAdminForm.ShowMsg();
        }
    }
    public static class ControlID
    {
        public static string TextData { get; set; }
    }
}
```