ECE 1776F Computer Security Graduate Project: Secure Electronic Elections

Alex & Philipp Hertel

December 8, 2004

Introduction & Motivation

Since the advent of democracy, the issue of election legitimacy has been crucially important. Illegitimate elections or even elections in which there is an appearance of impropriety undermine the democratic process and cause people to distrust their governments. One need not dig very deeply in order to find examples of controversial elections, even in industrialized countries. For example, the 2000 U.S. election, in which George Bush won, angered many voters and caused them to lose trust in the system. More recently, the 2004 U.S. election also saw its share of controversy, albeit not as severe, and the Ukrainian elections sparked massive street protests.

In order to help secure election legitimacy, many rules have been developed over the years. For example, most democratic countries implement at least the following protocols and regulations:

- Ensure that voting is anonymous. This allows people to vote as they choose, free of external coercion.
- Do not issue any kind of receipt which can allow someone to prove who they voted for. This counteracts the possibility of people 'selling' their votes, as well as the possibility of coercion.
- Ensure that voting precincts are staffed by members of the various parties running in the election. This allows them to act as checks against each other and the possibility of voting fraud.
- Enact laws which require officials governing elections that are or appear to be in conflicts of interest to recuse themselves.

There are many more such standard rules, and yet many elections continue to be controversial. As described in [1, 2], there has been much recent research into secure and verifiable electronic voting, and many of the results have been promising; it is possible to implement election systems that are much more secure and trustworthy than those currently used. Although none of these systems has yet been implemented and used in practice, it is almost certainly only a matter of time.

One important thing to note is that not all 'e-voting' schemes are equal, or for that matter, more than superficially similar. In fact, contrary to popular perception, the fact that many of the proposed cryptographic election systems would be implemented electronically is a largely irrelevant detail that provides speed of computation and convenience to the voter, and has little to do with security, verifiability, or trust. The underlying system ensures the election's legitimacy; the electronic implementation is simply the medium which allows voters to easily use it, provided that it accurately embodies the underlying system.

It is very easy to envision electronic voting systems which would provide the voter with no assurances of the election's legitimacy, even if perfectly coded. For example, the recent 2004 U.S. election featured electronic voting in the form of 'Diebold Machines', electronic voting machines made by Diebold Inc. The use of these machines was highly controversial. Not only have there been allegations of insecurity in the form of easy vote tampering, but the Diebold machines did not produce any kind of verifiable certificate, physical or otherwise, that would inspire trust in the system.

Given an unverifiable system like this, it becomes easy and indeed justifiable for voters to mistrust the election's result, especially since the president of Diebold Inc. in a 2003 letter told Ohio Republicans that he was "committed to helping Ohio deliver its electoral votes to the president next year" [5]. Regardless of whether any fraud was committed using the machines, at the very least the appearance of impropriety and lack of verifiability was enough to shake the confidence of many voters in the election's legitimacy. Had the results tallied by the Diebold machines been transparently verifiable by any citizen, then no political discourse surrounding the vote could have undermined the people's trust.

The purpose of this project is to implement a secure election system and to explore its security with respect to voter anonymity, receipt-freeness, public verifiability, and robustness against a coalition of malicious authorities. In addition, we will explore which assumptions of trust the various parties taking part in the election are required to make under the context of our implementation.

Project Description

Non-Electronic Version

Before describing our implementation, it is useful to describe an equivalent voting system that is not reliant on electronics or any other technology. This abstraction will allow us to emphasize the fundamental nature of the system without becoming mired in technical details. Once the nature of the system is clear, we will describe the design decisions we made as well as the more interesting implementation details. The general idea for this election system was adapted from [2].

Structure

Democratic elections often require a hierarchical structure that is based on geography. The details vary from country to country, but in general it is impossible for all people to vote at one poll / precinct, so at the very least it is necessary to have a number of polling stations distributed geographically. This gives rise for the need to organize these polls in some way. The standard way to do this is to have a hierarchical tree structure. For example, our project follows the U.S. Federal electoral system in which each state is divided into counties, each county is divided into precincts (church basements, school gymnasiums, etc.), and each precinct contains a number of voting booths. As shown below in Figure 1, each entity or 'authority' forms an internal node in the hierarchical tree, and the voting booths through which voters interact with the system form the leaves.

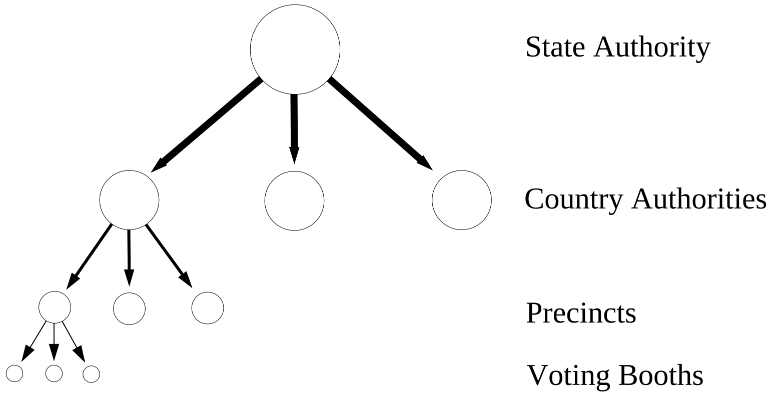


Figure 1: The Tree Structure Underlying The U.S. Federal Election System

Note that it is not necessary to have a node at the national level, since the electoral college inherently handles this abstraction. We need therefore only set up one voting system for each state. The fact that we are modelling the U.S. system is irrelevant; it is easily adapted to serve any electoral system based on a tree structure having a depth of at least three.

Goals

Again mirroring the U.S. system, one obvious goal is to be able to accurately tally votes at each level of the hierarchy. However, there are more goals which affect the legitimacy of the election:

- Votes must be anonymous.
- There must be no receipts linking people to how they voted.
- For legitimacy, the election should be publicly verifiable.
- The system should be robust; any fraud should require a large conspiracy.

Non-Technical Implementation

We assume that a voter registry exists such that every voter is uniquely identifiable (possibly by Social Security number) and has a secret password. In addition, we assume that each voter has an RSA public / private key pair for the purposes of signing. Voters keep their private keys hidden, and publish their public keys.

Each authority (state, county, and precinct) generates a Paillier public / private key pair, and publicly displays an enormous bulletin board as well as their public key. The private key is split in an intelligent manner amongst t people chosen from opposing parties. This is called 'threshold cryptography', and guarantees that decrypting anything that was encrypted using an authority's public key requires all t key fragment holders to work together, making malicious collusion difficult.

Let n be a loose upper bound on the population of the state (ie. $n >$ population size). Each of the j candidates is given a unique number i in $[1, j]$, and is given the unique identifying number $k_i = n^i$. This will allow us to add all of the numerical values representing votes together, and once finished, look at the result in order to discern the tally. For example, if our state population is 9, and we have 3 candidates, A, B, and C, we choose $n = 10$, assign $k_1 = 10^1$ to candidate A, assign $k_2 = 10^2$ to candidate B, and assign $k_3 = 10^3$ to candidate C. Let's say 1 person votes for A, 6 people vote for B, and 2 people vote for C. The summation S of all the vote values is $S = 1 \cdot 10^1 + 6 \cdot 10^2 + 2 \cdot 10^3 = 2610$. This number can be unambiguously decoded to show that candidate C received 2 votes, candidate B received 6 votes, and candidate A received only 1 vote. Because of the bound on our population, we know that it is impossible for one candidate's votes to overflow into the digits of another candidate and thereby invalidate the results of the election. This method generalizes to any population size.

Voting proceeds as follows: voters go to their precincts, log in at the entrance by presenting their unique identification and password, at which point they are checked off of the voter list, and allowed to enter. Each voter encrypts three times the number k_i associated with the candidate that he or she wants to vote for, once using the precinct key, once using the county key, and once using the state key. Each is then signed using the voter's private RSA key. The voter then proceeds to the bulletin board, and writes his / her unique id (so that the signature can be checked) as well as the three ciphers into a designated slot. Note that since Paillier is randomized, two votes for the same candidate have an asymptotically small probability of encrypting to the same cipher, so no two voters from the same precinct, region, or state will produce the same ciphertexts, even when cast for the same candidate.

Once the voting is finished, the tallying phase of the election begins. By using each voter's public RSA key, each precinct immediately verifies via the signature that the votes came from the voters that they were supposed to come from, and removes the signatures. The tallying then proceeds as follows: Let m be the number of voters that voted at the precinct. Let $x_1, x_2, ...x_m$ be the votes encrypted using the precinct key, let $y_1, y_2, ...y_m$ be the votes encrypted using its parent's (county) key, and let $z_1, z_2, ...z_m$

be the votes encrypted using its grandparent's (state) key. Each precinct calculates $X = \prod_{i=1}^{m} x_i$, $Y = \prod_{i=1}^{m} y_i$, and $Z = \prod_{i=1}^{m} x_i$. These steps are obviously verifiable by the public. At this point it is important to mention a vital property, namely that Paillier encryption is *homomorphic.* That is, for any two sequences of integers $k_1, k_2, ..., k_j$, and $c_1, c_2, ..., c_j$, $E_{Paillier}(c_1 \cdot k_1 + c_2 \cdot k_2 + ... + c_j \cdot k_j) = E_{Paillier}(k_1)^{c_1} \cdot E_{Paillier}(k_2)^{c_2} \cdot ... \cdot E_{Paillier}(k_j)^{c_j}$. In other words, multiplying the ciphertexts of the votes together and then decrypting them will yield the same result as if we would simply have added all of the plaintexts. Therefore, even though X, Y, and Z are all products, they are all equivalent to $E_{Paillier}(S)$, the ciphertext of the sum of all of the unique candidate numbers over the votes cast in the precinct. Therefore, if we decrypt a precinct's X value, we will be left with S, which can easily be decoded to reveal the vote tally in that precinct. Note that no individual vote was ever decrypted so voters have total anonymity.

In order for a precinct to decrypt its X value, its t key fragment holders use a "combiner" machine that takes X as well as all key fragments as an input, outputs $D_{Paillier}(X)$, and immediately clears its memory. Briefly, the threshold cryptography works as follows: when dividing the key among the t keyholders, we fragment the key into t arbitrary pieces. These pieces become the coefficients of a degree-$(t-1)$ polynomial. A degree-$(t-1)$ polynomial is uniquely defined by at least t points. Find t suitable points on the curve of this polynomial, and give each keyholder the (x, y) coordinates of one of these points. This is that keyholder's key fragment. Even if $t-1$ of the keyholders collude, they still won't be able to find the coefficients of the polynomial because an infinite number of degree-$(t-1)$ polynomials pass through any $t-1$ points. To rebuild the key, the decryption machine simply reconstitutes the polynomial by solving a system of t equations with t unknowns and concatenates its coefficients. No human being ever has access to the private key, either during fragmentation or combination.

In order to tally the votes at the county levels, each precinct sends its S value, as well as its Y and Z products to its parent authority. Once a county authority has received these totals from each of its precincts, the county adds all of the S values together (the S value addition is clearly easily verifiable), and multiplies the Y values together to get Y^*, multiplies its Z values together to get Z^*. Its t keyholders then come together and decrypt the Y^* value, at which point it is compared with the sum of the S values. If there is a discrepancy, then we can track down the fraudulent precinct(s) by decrypting each individual Y value and comparing it to its corresponding S value. Note that no anonymity is lost because we are decrypting products of votes rather than votes themselves. This will only be done in the event of fraud.

Tallying votes at the state level is done similarly using the decrypted (plaintext) Y^* values and comparing them with the decrypted product of all counties' Z^* values.

If we assume that voters are honest, but make no assumptions about the honesty of the authorities, then voters can have a high degree of trust in this system. Under these assumptions, all of our goals are met:

- **Anonymity:** Voters can trust that their votes are anonymous. It would take a conspiracy among t malicious authorities in order to decrypt a vote.
- **Receipt-Freeness:** There are no receipts, because all votes are encrypted using the public keys from the different authorities, so no voter can decrypt their vote in order prove that they voted in a certain way.
- **Public Verifiability & Robustness:** Every operation performed at the bulletin board is done publicly using public information and is therefore fully publicly verifiable except for the single decryption. For example, every individual voter can check that their vote was actually counted included in the cipher product by reproducing the multiplication of the ciphertexts. Public verifiability is clearly built into the system at each level. The only flaw is the single decryption. At any

single level, a fraudulent combiner could give an inaccurate result. The hierarchical structure of the system guards against this occurrence since every level of the hierarchy acts as a safeguard against fraud in any lower levels; any discrepancy between tallies is immediately visible and traceable. Fraud would require a conspiracy involving all t keyholders in every authority along one path down the tree, from the precinct level all the way up to the state level or colluding faulty combiners at each level. Not only is the first scenario highly unlikely, but it is doubtful that any system could guarantee legitimacy under these conditions. The second scenario can be guarded against by using the combiner a set number of times, once on the real product and a number of times on known ciphertext, plaintext pairs, input into the machine in random order. The combiner will have no way of knowing which input already has a known output and will therefore not be able to change the result.

However, there is one major problem with this system. The conclusion that this system meets our goals is dependent on the assumption that voters can be trusted. Unfortunately, it is very easy for voters to commit fraud in the following way: after entering the precinct and choosing a candidate's number n^i, instead of encrypting that number, a voter can encrypt $5 \cdot n^i$ instead, thereby making his vote count five times. Nobody can see the value of the vote, so it is easy for a voter to make their vote count an arbitrary number of times. This clearly undermines the entire election. There are a number of solutions to this problem. We can set up a booth that forces the voter to choose one of the numbers $n^1, n^2, ..., n^j$, and then encrypts that vote automatically, thereby removing any voter's ability to commit this type of fraud. However, this raises another problem; the voter now needs to have trust in the voting booth. Another solution is to leave the encryption in the hands of the voter so that it can be trusted, but to require the voter to provide three zero-knowledge proofs along with the encrypted votes to the precinct-level bulletin board. Precincts will only include votes whose zero-knowledge proofs verify that they encrypt the same legitimate value from $n^1, n^2, ..., n^j$. This requires the voter to have faith in the bulletin board's zero-knowledge proof verifier. An interested reader should refer to [2, 4].

Implementation & Design Decisions

Our implementation of the election system described above consists of a number of programs which interact to form the system. In order to solve the voter trust problem, we decided to implement voting booths that perform all of the encryption rather than leaving the encryption up to the voter, and requiring a zero-knowledge proof. This design decision was made for two reasons; firstly, implementing a zero-knowledge proof system together with a verifier is beyond the scope of this project in that it would have required considerably more time.

The programs comprising our implementation of the election system are as follows:

- The Voter Register: This program maintains a list of registered voters, their unique identification values, their passwords, as well as their public and private keys. In reality, voters should be in exclusive control of their private keys, but since we did not want the user to have to input numbers containing hundreds of digits, we included the private keys in the register. This was done purely for user-friendliness for the classroom presentation, and was not necessary, so should not be viewed as a potential breach of trust.

- The State Authority: This program maintains the state's bulletin board, tallies the information sent to it by the county authorities, dictates when the election is over (which initiates the tallying process), and announces the winner of the election. In addition, the state authority program ensures that all authorities below it in the hierarchy are legitimate and that their tallies can be trusted. Only one copy of this program may be running at any one time.

- The County Authority: This program maintains a county's bulletin board and tallies the information sent to it by the precinct authorities. County authorities are responsible for communicating their results to the state authority. In addition, each county authority program ensures that all of its children in the hierarchy are legitimate and that their tallies can be trusted. Any arbitrary number of instances of this program may be running during the election.

- The Precinct Authority: This program maintains a precinct's bulletin board and tallies the information sent to it by the voting booths. Precinct authorities are responsible for communicating their results to their parent county authorities. Any arbitrary number of instances of this program may be running during the election.
- The Voting Booth: This program controls a voter's login, presents the voter with the candidates, allows the voter to select a candidate and cast one vote. Each booth program performs all encryption and signing for every vote that is cast. Once the vote has been cast, the booth program reports to the register program so that a voter cannot vote again. Votes are sent to parent precinct authority programs. Each precinct program may have an arbitrary number of booth program instances running during the election.

It should be noted that all communications between programs are encrypted using SSL 3.0, making a man-in-the-middle attack very difficult to carry out.

Another design decision worth mentioning is that in the context of our implementation, it made little sense to include the t keyholders at each authority level. In a real voting system, breaking the key into t pieces would be vital, and storing these pieces in one place would be a serious security breach. Since we are simulating the election system and don't have t volunteers that could act as keyholders at each authority, we have no choice but to store the authorities' private keys, fragmented or not, in memory. Implementing threshold cryptography in this context would have been moot. A combiner class has been included which could readily be fleshed out to fragment and recombine the private keys.

We also decided to implement the variant of the Paillier Cryptosystem described in [3], rather than Paillier's original system. This is due to concerns with the original system's efficiency when implemented.

This project was implemented using Java, and has been tested under Windows XP. Java has a large library which saved much programming time. For example, Java has a BigInteger class which is useful when working with the extremely large numbers encountered when using cryptography, and many algorithms and protocols such as SSL 3.0 are provided by its libraries. In addition, Java is highly portable to different platforms and allows applications to be easily put on-line by using Java applets. This allows our voting system to be easily extended into an on-line version.

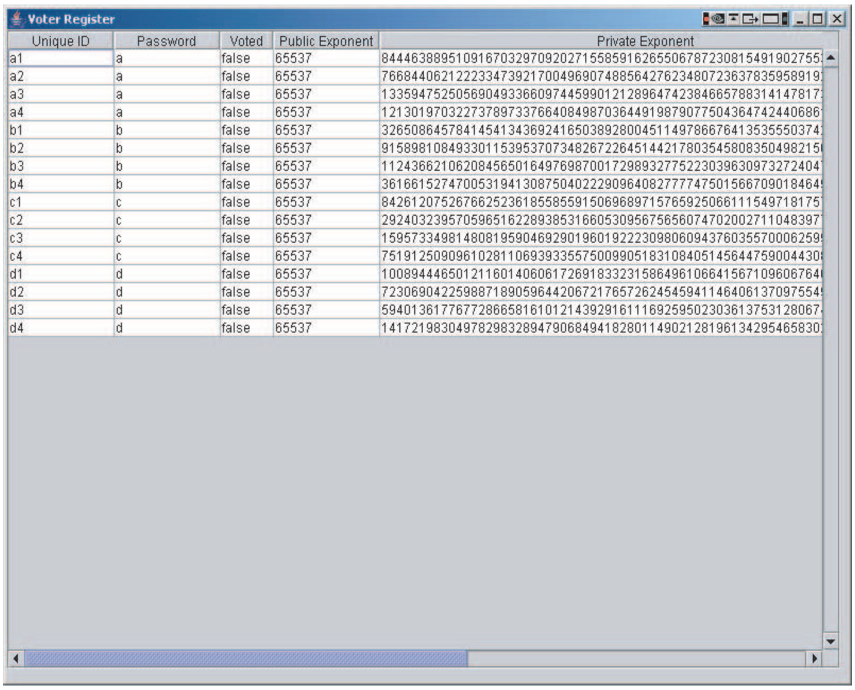
Example Execution of The Voting System Implementation

The Java project implementation has a graphical user interface, and as mentioned before doesn't consist of a single program, but rather consists of a coalition of programs working together over a network. To illustrate the functionality of the voting system, it would be useful to step through the following example execution:

Step 1: Getting The System Up & Running

For the purpose of this example, we will set up a system that consists of one voter registry, one state authority, two county authorities, and four precincts, two for each county. Each precinct has one voting booth. We assume that this example execution is being run under Windows, that Java has been included in the system path, and that all programs are being run on the same computer. If more than one computer is desired, the relevant IP addresses must be changed in the networking code.

Go to the directories containing the programs, and double-click run.bat in the following order: the voter register, the state authority, the regional (county) authority (twice), the local (precinct) authority (four times), and the election booth (four times). The system is currently hard-coded so that each authority in the tree can only have a maximum of two children. This can easily be adjusted by changing a constant in the various XXXXServer.java files and re-compiling. The programs will automatically communicate and synchronize with one another. Note that the voter registry already contains a list of voters as well as their passwords and keys, as shown in Figure 2 below.



Unique ID	Password	Voted	Public Exponent	Private Exponent
a1	a	false	65537	84446388951091670329709202715585916265506787230815491902755
a2	a	false	65537	76684406212223347392170049690748856427623480723637835958919
a3	a	false	65537	13359475250569049336609744599012128964742384665788314147817
a4	a	false	65537	12130197032273789733766408498703644919879077504364742440686
b1	b	false	65537	32650864578414541343692416503892800451149786676413535550374
b2	b	false	65537	91589810849330115395370734826722645144217803545808350498215
b3	b	false	65537	11243662106208456501649769870017298932775223039630973272404
b4	b	false	65537	36166152747005319413087504022290964082777747501566709018464
c1	c	false	65537	84261207526766252361855855915069689715765925066111549718175
c2	c	false	65537	29240323957059651622893853166053095675656074702002711048397
c3	c	false	65537	15957334981480819590469290196019222309806094376035570006259
c4	c	false	65537	75191250909610281106939335575009905183108405145644759004430
d1	d	false	65537	10089444650121160140606172691833231586496106641567109606764
d2	d	false	65537	72306904225988718905964420672176572624545941146406137097554
d3	d	false	65537	59401361776772866581610121439291611169259502303613753128067
d4	d	false	65537	14172198304978298328947906849418280114902128196134295465830

Figure 2: The Voter Register Program

Step 2: Holding The Election

All voter interaction with the system is carried out through the election booth programs, shown below in Figure 3. Voters log in using their unique IDs and passwords, which are verified with the voter registry. Each voter then clicks on the candidate of their choice, and selects the circular 'vote' button. A dialog box opens asking the voter to verify their choice. Once a vote has been cast, the booth resets for the next voter to log in.

Let us simulate the 2004 U.S. presidential election in which the three primary candidates were Ralph Nader, John Kerry, and George Bush. Using all 16 voters in the registry, send some of them to each precinct, and cast a vote. The booths correspond to the precincts in the order that they were executed. Each time a vote is cast, it will appear in encrypted form on the precinct bulletin board.

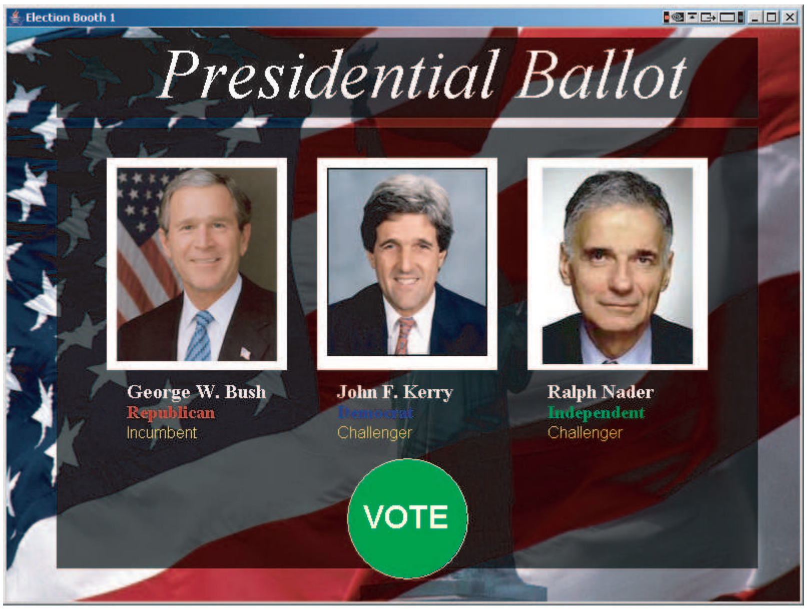
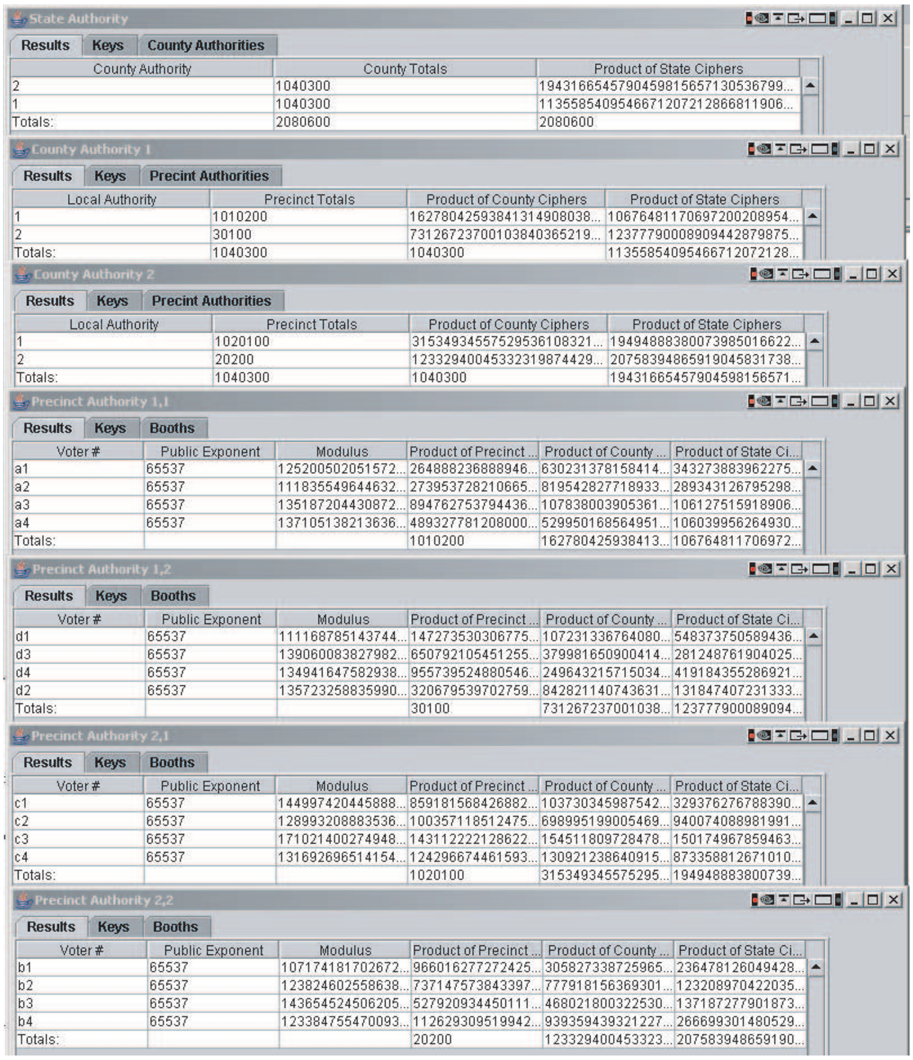


Figure 3: The Election Booth Interface

Step 3: Tallying The Vote

Once the balloting is finished, it is time to tally the vote. Go to the state authority window, and click on the 'tally' button. This causes all vote information to be passed up the tree. All multiplication and decryption is carried out automatically at each authority. Figure 5 below shows the bulletin boards from a sample election. Note that we assume that the state population is no greater than 99, so we choose $n = 100$. George Bush's associated value is $k_1 = 100^1$, John Kerry's associated value is $k_2 = 100^2$, and Ralph Nader's associated value is $k_3 = 100^3$. The unencrypted tallies shown on each authority's bulletin boards can easily be decoded to show that the votes were tallied properly.



State Authority

County Authority	County Totals	Product of State Ciphers
2	1040300	1943166545790459815657130536799...
1	1040300	1135585409546671207212866811906...
Totals:	2080600	2080600

County Authority 1

Local Authority	Precinct Totals	Product of County Ciphers	Product of State Ciphers
1	1010200	16278042593841314908038...	10676481170697200208954...
2	30100	73126723700103840365219...	12377790008909442879875...
Totals:	1040300	1040300	11355854095466712072128...

County Authority 2

Local Authority	Precinct Totals	Product of County Ciphers	Product of State Ciphers
1	1020100	31534934557529536108321...	19494888380073985016622...
2	20200	12332940045332319874429...	20758394865919045831738...
Totals:	1040300	1040300	19431665457904598156571...

Precinct Authority 1,1

Voter #	Public Exponent	Modulus	Product of Precinct ...	Product of County ...	Product of State Ci...
a1	65537	125200502051572...	264888236888946...	630231378158414...	343273883962275...
a2	65537	111835549644632...	273953728210665...	819542827718933...	289343126795298...
a3	65537	135187204430872...	894762753794436...	107838003905361...	106127515918906...
a4	65537	137105138213636...	489327781208000...	529950168564951...	106039956264930...
Totals:			1010200	162780425938413...	106764811706972...

Precinct Authority 1,2

Voter #	Public Exponent	Modulus	Product of Precinct ...	Product of County ...	Product of State Ci...
d1	65537	111168785143744...	147273530306775...	107231336764080...	548373750589436...
d3	65537	139060083827982...	650792105451255...	379981650900414...	281248761904025...
d4	65537	134941647582938...	955739524880546...	249643215715034...	419184355286921...
d2	65537	135723258835990...	320679539702759...	842821140743631...	131847407231333...
Totals:			30100	731267237001038...	123777900089094...

Precinct Authority 2,1

Voter #	Public Exponent	Modulus	Product of Precinct ...	Product of County ...	Product of State Ci...
c1	65537	144997420445888...	859181568426882...	103730345987542...	329376276788390...
c2	65537	128993208883536...	100357118512475...	698995199005469...	940074088981991...
c3	65537	171021400274948...	143112222128622...	154511809728478...	150174967859463...
c4	65537	131692696514154...	124296674461593...	130921238640915...	873358812671010...
Totals:			1020100	315349345575295...	194948883800739...

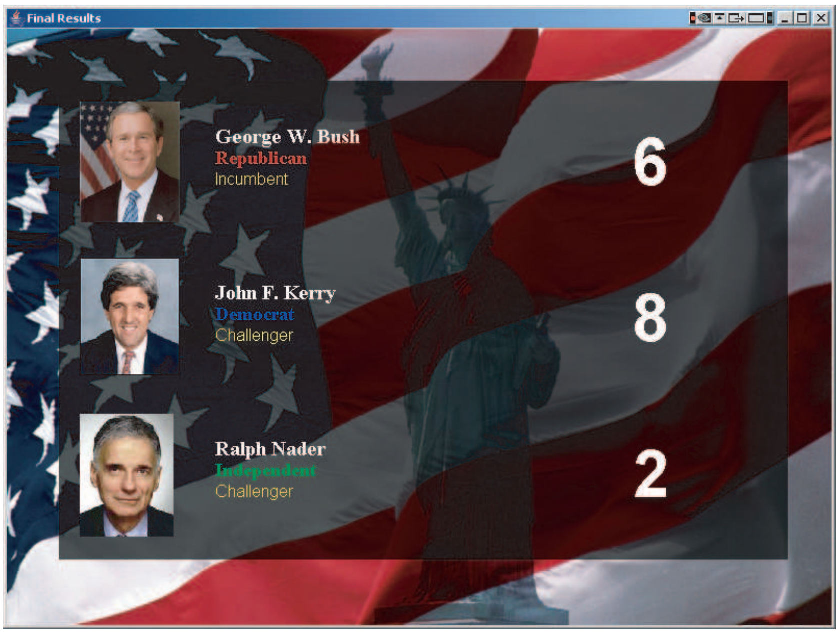
Precinct Authority 2,2

Voter #	Public Exponent	Modulus	Product of Precinct ...	Product of County ...	Product of State Ci...
b1	65537	107174181702672...	966016277272425...	305827338725965...	236478126049428...
b2	65537	123824602558638...	737147573843397...	777918156369301...	123208970422035...
b3	65537	143654524506205...	527920934450111...	468021800322530...	137187277901873...
b4	65537	123384755470093...	112629309519942...	939359439321227...	266699301480529...
Totals:			20200	123329400453323...	207583948659190...

Figure 4: Authority Bulletin Boards

Step 4: The Election Results

Once the tally is complete, the election results screen, shown below in Figure 5 will automatically pop up, showing the results of the election.



Final Results

George W. Bush
Republican
Incumbent
6

John F. Kerry
Democrat
Challenger
8

Ralph Nader
Independent
Challenger
2

Figure 5: The Election Results Screen

Discussion and Conclusion

We believe that this implementation has been able to successfully illustrate many of the issues and subtleties surrounding the design of a secure voting system that has requirements such as anonymity, receipt-freeness, trust, and robustness. As such, this project constitutes a successful 'proof of concept'. At the very least, it shows that it is possible to set up an election system that is more credible and inspires more trust than the current U.S. system.

We believe that this system is ready to perform a real-life election, using any number of candidates and voters. Minor changes would naturally have to be made, such as changing the pictures and names of the candidates etc. and obtaining a proper list of eligible voters. The only real concern is whether voters would trust our booth. Given the addition of the zero-knowledge proofs and informed voters, this problem could easily be overcome. Each voter could simply be responsible for sending his or her vote to the bulletin board using whichever implementation of a booth that they do trust. As long as the vote has the correct form, three signed ciphers and the accompanying zero-knowledge proofs, the bulletin board could trust that the vote is legitimate and count it. Our implementation is a good first step and is ready to have the zero-knowledge proofs and verifiers included.

Being theory students, we do not get many opportunities to program, so a lot was learned during the implementation process which could be used to create a better implementation in the future with respect to considerations other than security. For example, it would be a good idea to set up the booth in such a way that its ballot could be easily changed, perhaps by using an HTML page as the ballot, or even having a special class containing all of the constants used by the different programs.

Clearly it is possible to use technology in order to enhance election legitimacy. Furthermore, it stuns us that two busy graduate students were able to code a system in a few weeks which comes very close to being truly trust-worthy, while large corporations and even nations are not able to do so given years, large budgets, and incredible manpower. This seems to demonstrate a lack of interest on their parts.

References

[1] A. Acquisti. Receipt-Free Homomorphic Elections and Write-in Voter Verified Ballots. *CMU-ISRI-04-116*, 2004.

[2] O. Baudron, P. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System. *PODC (ACM)*, pages 274–283, 2001.

[3] Ivan Damgard, Mads Jurik, and Jesper Buus Nielsen. A Generalization of Paillier's Public-Key System with Applications to Electronic Voting. *PKC 2001*, 2001.

[4] O. Goldreich. Zero-Knowledge Twenty Years After Its Invention. Technical Report, 2002.

[5] Julie Carr Smyth. Voting Machine Controversy.
Published On Internet: http://www.commondreams.org/headlines03/0828-08.htm.