INNO.VOTE

SOFTWARE THAT POWERS DEMOCRACY SHOULD BE FREE.

WHITE PAPER DOCUMENTATION

Prepared for: Innovote Solutions, Inc.
Prepared by: Stefan Nagey
Version 1.1 / June 01, 2016

The Inno.vote Mission

Inno.vote aims to bring accessible, secure, transparent, and verifiable elections mediated by the Internet to governments everywhere at a price point an order of magnitude less than current commercial providers.

The Inno.vote suite of products and services will be able to operate alongside legacy providers, or will be able to provide a complete replacement suite of software. Further, legacy systems can be made compatible with Inno.vote's BallotChain technology.

Inno.vote's long-term goal is to create a trusted and verifiably secure system for unsupervised remote voting.

We will bring these goals to life with the SecureVote suite of elections software and the Inno.vote Elections as a Service (EaaS) platform.

Accessibility

Anyone who wants to vote, and has the legal right to vote, should be able to vote.

Security

Voters should trust that their systems of government are secure from tampering, whether that tampering should originate from a foreign power, domestic interest, or elections official.

Transparency

Functional democracy requires that elections be transparent, that information about contests be readily available, and that ballots cast be kept secret, yet available for inspection.

Verifiability

The public has the right to verify every piece of an election and submit any outcome to scrutiny, with the exception of knowing for whom any individual has voted.

Price

Democracy is a public good, not a profit center, therefore the burden on a jurisdiction to facility democracy should be minimal.

Inno.vote's model is based on providing Elections As A Service (EaaS), providing a complete suite of software and services for local elections authorities to mediate elections using the SecureVote suite of elections management software.

Elections authorities have a mandate under the Help America Vote Act of 2002 (HAVA) to integrate with numerous other state and local agencies to make voter registrations easier and more complete. Inno.vote will provive a suite of integrations and integration services for these legacy systems and databases with the SecureVote suite of elections software.

Commodity Hardware

All SecureVote project software is designed to run on commodity hardware. The voting client runs on iOS (and eventually android), all other software is written in mainstream languages that will run on POSIX-compliant server hardware. This will dramatically bring down the cost of implementation for jurisdictions as it will remove the need custom hardware provided by legacy elections providers that is costly to acquire and maintain.

Open Source Software

Inno.vote will be the primary contributor and maintainer of SecureVote projects, chief among them, the SecureVoter ballot specification, and SecureVote Pollbook, Elections Management System (EMS), Voting client, SecureVoter Tabulator, and Registration databases. Any elections authority or concerned citizen may contribute to these projects, and anyone may run the SecureVote Project's software on their own hardware.

The SecureVote Standard — A Secure, Transparent and Open Scheme to Conduct Supervised Elections

SecureVote is a series of open source software projects and standards that provide a method to conduct elections with supervised and remote voting components. The project will include the following major components: voter registration database, elections management system, poll book, tabulator, voting booth iPad application and booth schematic, and voting client for iPhone and Android. The project will provide standards for ballot style definitions, encryption protocols for ballots cast in an election, and public auditing procedures for elections systems.

Security through Open Source Software

All current voting systems software in use in the United States is closed-source software produced by private firms. While these firms do submit their source code to elections authorities to review, these authorities are neither security nor computer science experts. Having open, global review of software makes for far greater security and trust of a system overall.

Software can have multiple vulnerabilities, backdoors and bugs. These can be introduced by careless or disgruntled developers, or can be put in place through malicious action or malfeasance. Closed source software has fewer opportunities to find and review these types of issues, though internal QA, government review and testing labs do help remediate this risk. Inno.vote's SecureVote open source voting software will go through all of these same internal

and government checks, but will have the added layer that any citizen, any voter, any security researcher, any academic, any enthusiast can also review and comment on the code. This is a proven, tested model that underpins nearly all modern software. Google, Apple, Facebook, and even Microsoft build their businesses on Open Source Software.

While no single solution or technique can be a panacea for software issues, the increased review made possible by the Open Source approach is the best way that the software engineering community has come up with to develop robust software so far. The software that underpins freedom must be free.

Security through Public Key Encryption Standards

SecureVote uses a series of publicly available, high-quality strong encryption schemes. Among these schemes are RSA, Threefish, SHA256, SHA512, and ElGamal. A complete listing of security protocols used in message security can be found in Appendix B.

Maintaining Ballot Secrecy

The secrecy of the ballot is key to a functioning democracy. The SecureVote standard provides for multiple layers of ballot security, while maintaining accessibility, transparency and verifiability. To fully understand the security protections of the SecureVote standard, it is important to understand the principles of cryptography as laid out in the appendix to this document.

In general terms, the SecureVote standard in effect creates a ballot box for each individual ballot, allowing that ballot to be transported and copied by untrusted sources. Each voting device will lock each ballot in an individual digital ballot box and then seal that box with a digital tamper-proof seal.

The key to unlock these boxes is known only to the elections authority. There is a separate key to lock the boxes that is freely copied. The key used for locking cannot be used for unlocking.

Protecting Against Voter Coercion

Voter coercion can come from three major sources. The first such source is physical coercion. The next source comes from voters having some method by which it can be determined for whom they have voted, thereby breaking the secrecy of the ballot. The third source is some method by which someone looking to coerce voters (the “attacker” hereafter) can determine that specific votes have been cast through having access to all ballots cast. This is most easily accomplished by the attacker making the voter write-in a specific code word or phrase for a race in which the attacker is disinterested. It also represents a fourth attack vector as it can be accomplished by having voters encode information in various preference / ranked choice voting

races. For example, a typical local judicial or school board race race with eight candidates in which a voter has to pick six could have 20,160 *(8 nPr 6)* possible messages encoded in a vote.

The first type of coercion is the easiest to protect against in a supervised election, but the most difficult in cases of remote voting. In a supervised election, the polling place and elections workers are the method of preventing coercion. For mobile voting, a voter will be able to re-vote their ballot as many times as they wish during the election window. Each new/replacement ballot that is cast is accompanied by a ballot trump that is signed by the voting device. The ballot trump invalidates older ballots and gives the ID of the newest ballot. The presence of a ballot trump requires the tabulator to collect all ballots signed by a particular voting device, invalidating any ballot that has been indicated in a ballot trump. In the case that no valid ballot can be found (indicating a dishonest voting device), the original ballot (the ballot which was trumped, but did not cause any trumps) will be counted.

We protect against the third type of coercion through our system of HashIDs for write-in candidates by assigning a unique HashID to each write-in vote, and then aggregating these HashIDs into candidates during the aggregation process. During this aggregation process, the HashIDs that correspond to write-in candidates will be released by the elections authority if and only if the number of votes for that candidate has crossed a certain threshold, usually 1,000 votes, or 1% of all votes cast, whichever is smaller. While it is true that from this information a group could still determine if a large number of people had voted according to their instructions, this is information that could be learned from the vote totals regardless.

The final type of coercion cannot be solved without splitting the ballot into separate artifacts. In this case, the ballot token will contain multiple ballot ids, and each ballot created by the voting device will either vote all simple contests and undervote all ranked choice contests, or will contain a vote for a single ranked choice contest, and undervotes for all other contests. Additionally, each of the ballots will have a random time-delay applied before being fully released, such that an attacker cannot use time or sequence to correlate individual ballots to construct a complete record of a vote.

Verifiable Vote Totals

This system optionally makes vote totals externally verifiable by releasing copies of all ballots in the system to trusted observers, or beyond that, making copies of all ballots available to general observers. Ballots are released encrypted, in real-time. As each ballot contains a timestamp when it leaves the voting device, observers can record the time at which a ballot was received to obtain a measure of certainty that a ballot was not tampered with or delayed in transit. It bears noting at this time as well, that while a ballot could be tampered with in theory, doing so would require corrupting the voting device **and** the central voting authority. There are various methods to protect against corruption detailed in the following section.

Preventing and Detecting Corruption

Corruption in this section is defined as a dishonest software agent. Accompanying each release of an SecureVote project is a disk image which contains the full stack software required to run the project, alongside a checksum. For any election facilitated by Inno.vote's EaaS offering will provide verification of all system components at the beginning and end of the election. Device management access (i.e., via ssh or similar) is specifically disallowed on devices in the EaaS cloud, and each device managing an election lives only long enough to manage the election in question. Together, these measures make tampering with an election without detection impossible, no matter the level of access available to the attacker.

At every level of the system beyond the individual voting system, redundancy is built in, such that if an attacker were able to corrupt one facet of the system, that corruption would be immediately detected by other agents within the system, and the corrupted system removed. While this could cause logistical difficulties with a sufficiently large and coordinated attack, at no time could an attacker compromise the results or validity of an election.

Verifying the Presence of an Individual Vote in the Totals

It is most helpful to think of the receipt section of a SecureVote ballot as a tear-off receipt of a paper ballot. Each ballot has multiple receipts, one to be torn off and recorded by each tabulator compiling results of an election. Each receipt contains an encrypted HashID. After a tabulator has validated a ballot and included it in its tabulation, it will publish to its valid ballot list (for public inspection) the SHA512 hash of the entire ballot, and will also record the decrypted receipt value, so that a voter can use their ballot receipt (which contains a HashID) to confirm that their ballot was included in the tabulation performed by that tabulator.

Since HashID ballot receipts cannot be guessed, the user can determine if their vote was included in the totals, but is unable to determine the inclusion of any other specific votes for which they do not possess the receipt.

This operation requires both security access and trust of the parties with access. Each SecureVote ballot includes a section of receipts. Each of these receipts is encrypted for a specific authorized tabulator. These tabulators will include the precinct in which the vote was cast (for supervised votes), the elections authority, as well as any trusted third-party observers (e.g., the UN, NDI, etc). The private keys to decrypt these receipts are to be kept private to the authority to which they belong, as they can be used to correlate a ballot with information that a voter possesses (their vote receipt).

Transparency of Elections

Central to the integrity of an election is the ability of its results to stand up to scrutiny at any level, and from any source. With Inno.vote and the SecureVote standards, elections are fully verifiable at every level.

Verification of Tabulations

Prior to the election, each “official” tabulator will publish the rules it uses to determine the validity of a ballot, the public keys of all authorized voting devices. After the conclusion of the election, the elections authority will publish the election’s private key such that ballots may be decrypted, and each tabulator will confirm the SHA512 hashes of ballots included in its tabulation. Each tabulator will also publish the SHA512 hash of any invalid ballot, as well as the reason for which it was rejected. Since all ballots cast are broadcast in an encrypted form as they are cast, an untrusted observer, using only this information will be able to confirm the following:

- The tabulator published accurate vote totals
- The vote totals published by all tabulators agree
- The ballots included by all tabulators are the same
- The tabulator included all valid votes for each valid voting devices
- The tabulator included only votes from valid voting devices

Observers will also be able to inspect the entire ballot cohort to determine any ballots cast from invalid voting devices, any ballots from valid devices not included in the totals, and that the invalid ballots were rejected for valid reasons.

Authenticity of Ballots

Ballots can be analyzed and determined to be authentic or fraudulent by any observer during or after an election. The **authenticity** of a ballot is distinct from the **validity** of a ballot. The former is a determination if the ballot was generated by an authorized device, whereas the latter is a result of various rules for validity applied by tabulators to determine if the ballot is valid in the election.

Any ballot can be validated by comparing the SHA512 sum of the ballot to the list of valid ballots published by a tabulator. Additionally, a ballot can be validated as having come from a verified voting device during the election. At the start of the election, the elections authority will publish all valid public keys for all valid voting devices. The private keys are known to the voting devices alone. These public keys can then be used to validate signatures of the encrypted SecureVote ballots generated by that device. In this way, observers may authenticate ballots as being authentic, regardless of their validity.

Provisions to Protect Tabulations During an Election

While all ballots are broadcast during an election, the broadcast value is of no use to someone wishing to determine the results of an election prior to its conclusion. All ballots are encrypted with an "elections key." The public key is known from the start of the elections so that voting devices can encrypt ballots, but the private key is held and protected until the conclusion of the election by the elections authority. This key is best kept offline, and the Inno.vote platform provides easy capabilities for the export and import of this key. Following the conclusion of the election, the private key is provided to the tabulators (and optionally the public), and the key is "revoked" — meaning that no further ballots are to be encrypted with this key.

Registration of Individuals for Unsupervised Voting

The biggest single obstacle to unsupervised voting is the registration of voters. Voter registration will continue to necessarily be done via postal mail or in person. Since the registration process is what ties a voter's public key to the voter's registration, allowing them to vote. This is necessary to confirm four key items: 1) the individual performing the registration is actually the individual to be registered, 2) the individual is eligible to be registered, 3) the individual is not registered already, and 4) the address where the individual is domiciled. The SecureVote registration database will provide registration services to be used by an elections authority, but should not be opened to direct public registrations (i.e., registrations from a website or app) without further integrations or customizations by Inno.vote or another SecureVote solutions provider. The basic Inno.vote mobile client includes enhanced registration capacities and can be easily customized for an elections authority.

It is important to note that in order to use a SecureVote voting client or app, the voter must be a SecureVote registered voter, and not just a registered voter.

Registering the Individual to Vote

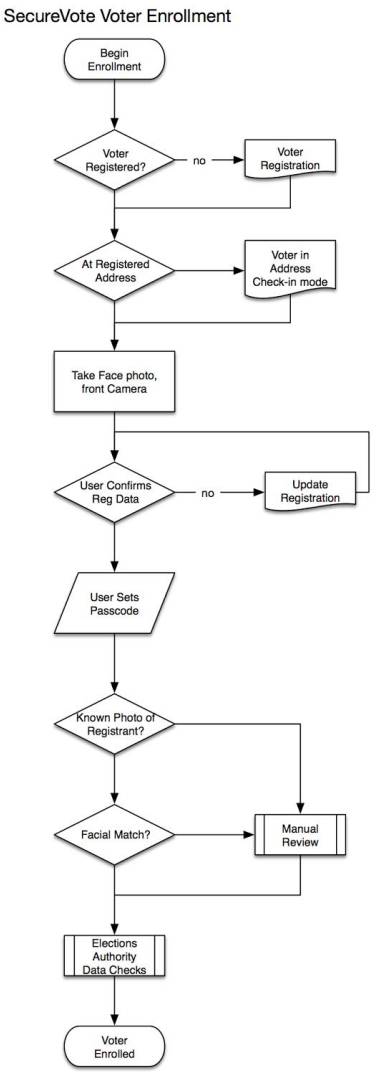
The SecureVote platform supports the import of legacy voter registration databases, but requires users to enhance their registration in order to use a SecureVote voting client or application. These enhanced registration characteristics are called SecureVote Enrollment.

For jurisdictions in states with driver's license databases that contain the license photographs, we can use provided information and facial recognition software (e.g., https://www.keylemon.com/oasis or https://mobile.bioid.com/) and a device's camera to photograph the individual and compare them

to the photograph associated with the license number provided. If there is a match, and the address given matches the driver's license database address, we can treat the registrant as fulfilling criteria #1 and #4.

In cases where a driver's license photograph is unavailable, whether the individual does not possess a license or the jurisdiction cannot provide the photographs, we will photograph the individual and accept their affidavit as fulfilling criteria #1.

Most jurisdictions require the signature of an affidavit only to fulfill criteria #2. For all jurisdictions, we will collect this signature at the time of registration. This will also be used when the voter signs the poll book at the time of voting, whether in person at a polling place, or through the app on their own device.



For jurisdictions that require a proof of citizenship, we can additionally photograph the registrant's face and passport photo page, verify that the individuals are the same (again, via facial recognition software) and use that to fulfill the citizenship requirement of criteria #2.

To fulfill criteria #3, we will compare the photograph of the individual's face to the database of faces for individuals already registered, as well as comparing the information provided to the information in the database. In the case where the individual appears to already be in the database with the name provided at the address provided, we can merge these records and treat criteria #3 as fulfilled.

For individuals for whom criteria #4 is not met through the driver's license database, we will photograph an acceptable "proof of residency" document (varies by jurisdiction). From this document, we will read (via OCR) the individual's name and address. If these match the information already provided, we can then reverse geocode the address to get a geographical position. If the voter is physically at the location indicated, we can treat criteria #4 as fulfilled. If the voter is not in the physical location at the time of registration, we can optionally require them to "check-in" at that location prior to the election in order to be fully eligible.

If an individual has fulfilled all four eligibility criteria, we can proceed to register them in the database. The registrant will select a password that will be used in the future to verify identity (in combination with their picture, their name and their address). The registration of an individual as a SecureVote voter in a registration database is distinct from the registration of a device to an individual, and the registration of a device in the database. At this point, we have confirmed that the individual is registered, and that they are the user of the device. At this point, they may continue to register their device.

Registration of a Device as Authorized to a User

For the registration of a device, a key pair (a public and a private key) must be generated. These can either be generated by the elections authority or by the voter's device.

We believe that the combination of physical location, biometric factors, and the integrity of the device and software as standard (so that the camera and physical location provided by the OS can be trusted) constitute a sufficient standard of proof that the user is accurately representing themselves and that user-generated keys may be used.

If the key pair is generated by the voter, then the voter's device will transmit encrypt the public key with the supervising authority's public key and transmit it to the supervising authority over SSL, where the authority stores it as a verified device belonging to that voter. If the key pair is generated by the elections authority, then the public key is stored immediately, and the private key is mailed through the postal service to the registrant as a series of QR codes that can be scanned with the app. Private keys should never be transmitted unless they are being simultaneously revoked.

The registrant's app on the device is then secured with a combination of a passcode and a biometric factor via the fingerprint sensors and/or facial recognition capabilities ubiquitous on mobile devices.

When the registrant wishes to vote, she must identify herself to her device with the biometric factor and the passcode. Then the private key is unlocked and is available to sign votes and other transactions.

Registering a New Device to a SecureVote Registered Voter

When a SecureVote registered voter wishes to move their registration to a new device, a new key pair must be generated. The user can identify themselves in the app with the combination of their registered name and address, password and live picture. At that point the new device must be registered as described above in the "Registration of a Device as Authorized to a User".

This presents the most significant challenge to jurisdictions who wish to generate the key pairs centrally as opposed to on user devices, as users get new devices more frequently than they vote.

The SecureVote Software Suite

The SecureVote project will implement not just a set of standards and protocols, but will also provide a complete suite of software to manage elections. All of these open source projects will be implementable and customizable per jurisdiction, and are all designed to work together to give provide for the complete management of elections.

Software will either be aimed at voters, elections officials, or systems.

Voter-facing Software

The voter-facing software will have the broadest footprint and will be the most important software to secure. This software will also be the one with which the public interacts (albeit infrequently), so user friendliness is of paramount importance.

Mobile Device Application

The mobile app will be the primary software through which citizens interact with the elections authority. The app will enable voter registration, voter registration updates and voting. The app will also provide for push notifications from the elections authority to voters reminding them to update party affiliations, register, as well as the opening and closing of voting periods. The app will also have functionality designed for poll workers. This app will run on iOS and Android.

Mobile device voters will have their ballots counted in the precincts where they are registered.

Voting Booth

The Voting Booth application will run in the polling place, and will be the system upon which the voter casts their ballots. This system will encrypt and transmit the ballots to the appropriate locations, and will print the ballot receipt for the voter.

This software will make use of the assistive features of iOS and run on iPads only.

Elections Official-facing Software

The software to manage the election will run via the web, and will be designed for elections officials to work with every day.

Registration Database

The registration database will maintain voter information and registration files and will be the canonical source for this information for all other systems and software. Each voter record will

have the basic identifying information, as well information on which elections a voter has been eligible for, and their participation (or nonparticipation) in those elections.

Election Management Software (EMS)

The EMS will manage elections, contests, parties, contestants and measures. The EMS also manages precinct boundaries and device registration (which devices are authorized to go with which precinct). These pieces of information go together to create ballot styles, which determine which questions get posed to a specific voter in a precinct.

The EMS also generates the election key pair, stores the private key in memory (to be the source of truth during the election), and will, at the end of the election invalidate and publish that private key to allow for tabulations.

Poll Pad

The Poll Pad software is used at the polling place by elections workers (not elections officials) to verify a voter's identity and sign them in as having cast their vote in the election. In the SecureVote system, the poll pad is only used to issue a voter a ballot token, and that no ballot token has already been redeemed for that voter. This system maintains a real-time accounting of all voters who have redeemed their ballot tokens (in other words, have cast their votes) to protect against double voting and other types of fraud.

Reporting Election Results (RER)

The Reporting Election Results (RER) software will collate the results from all tabulators and allow elections officials to review results from each precinct broken down by in person, absentee, early and mobile votes. This integrates closely with the EMS and tabulators to gather precinct data and collate the results from the election overall.

Systems Software

These internal software agents will manage various aspects of the process both during and after the election.

Integration Agents

Integration agents are small pieces of software that integrate any piece of the SecureVote system with an external database of any sort. For example, these could integrate the registration database with a DMV's database, the SSA database, or could integrate the precinct database with a streets or reverse geocoding database. These agents are also able to publish information as it changes within the system to any interested authorized parties.

Ballot Publisher

The ballot publisher will listen for ballots sent to it, verify the signatures of those ballots as being created by an authorized key for that election (whether created by a device enrolled by a registered user, or by a voting machine in the election), and if authorized, it will mark it as a “valid ballot”, re-sign it, and announce it to all registered listeners. It will also create a transaction to place it onto the public and private blockchains. Ballot publishers will run on each voting device and will be a part of the voting client.

Tabulator

The tabulator will accept a blockchain, the elections private key, and a collection of public keys from authorized machines, or from a trusted ballot publisher, and from that, it will decrypt and tabulate all ballots, producing a report of vote totals for all candidates in all contests, as well as a number of undervoted, trumped, and invalid ballots on the blockchain.

Audit Logger

Audit loggers subscribe to all events created during an election and log each and every one. These are multiple, fault-tolerant servers distributed between the cloud and owned locations, with at least two set to print every event. This creates a complete paper-based audit trail that can be made available for inspection by anyone in the elections authority or authorized external observer. Inno.vote will also offer this paper logging as a service to any jurisdiction, printing, indexing and collating all materials into books for later inspection.

BallotChain Technology

The voting blockchain or ballotchain will work as a distributed ballot box, or in cryptocurrency parlance, an altchain, being processed by all machines involved in an election, as well as external parties. There will be three copies of this database required for an election. First will be a copy being processed only by machines owned by an elections authority, second is a copy processed by all Inno.vote servers and all authorized external observers, and the third ballotchain is open to the general public. The purpose of these three different databases is to ensure data integrity and to protect against an attacker controlling 51% of any individual mining pool.

The genesis block of each election’s ballotchain will be the election’s public key, and each election will be finalized by publishing the private key as a transaction to the ballotchain. All ballots on the ballotchain between the public key and the private key will be considered valid if they contain a signature matching a valid ballot token published by the elections authority.

Each origination address on a ballotchain will correspond to an authorized voting client. Each ballot will be encoded onto the ballotchain as a transaction, establishing a timestamp for the ballot being cast as during the election. The content of the transaction will be a ballot encrypted using the public key from the ballotchain’s genesis block.

The address publishing the keys will be the election authority's address. This address will also publish transactions as new valid voting clients are added (and are assigned their own address). Ballot tokens are also published to the BallotChain from authorized BallotToken creation agents.

With a copy of a BallotChain, and observer can recreate every aspect of an election to verify its authenticity, validate the result and ensure that all transactions are signed, that all ballots match valid ballot tokens, and that no tampering has taken place in the form of forged ballots or tokens.

Selling Inno.vote and the SecureVote Standard around the World

Like any standard, SecureVote will be made stronger through adoption, revision and inspection. To this end, one of the primary goals of Inno.vote is the advancement of the standard, regardless of the company's sales. Our company's core mission is to improve elections.

Integrating SecureVote and the BallotChain with Other Voting Schemes

Inno.vote provides a complete suite of elections software, but it also plays well with others. SecureVote and BallotChain concepts can be seamlessly integrated with other voting schemes. Inno.vote is a natural choice for implementing UOCAVA voting, as it allows military and overseas voters to cast their ballots and maintain their ballot secrecy, something which current schemes do not provide for.

Inno.vote systems can work with legacy registration systems, can be deployed to certain precincts, can be deployed within precincts alongside traditional voting equipment, and can be used to handle overseas and absentee voting.

In split design systems, the Inno.vote vote totals and BallotChain will not be complete records by default, but traditional OMR systems can be retrofitted to write ballot contents to the BallotChain as they scan ballots, adding a layer of transparency and verifiabilty to elections systems already in place at zero risk to jurisdictions.

The State of Voting and Elections in the USA

American jurisdictions were largely burned by the first-generation of touchscreen voting devices, produced primarily by Diebold (now Dominion Voting Systems) and ES&S. As a result of these early failures, many jurisdictions have gone back to paper ballots tabulated by Optical Mark Recognition devices (OMRs), optionally marked by assistive marking devices. This system, while less likely to break down and require technical assistance on election day, is a step backwards in terms of technology. Even though the new tabulator heads have the functionality

to electronically transmit results to the elections authority, these technologies lag far behind the state of the art.

Many jurisdictions already have “no excuse” absentee voting and extended early voting periods of two weeks or more. These changes allow voters to get to the polls more easily, and fit voting more into the rhythm of daily life. Jurisdictions with these changes will be more readily able to accept mobile device voting systems.

The primary concerns in elections in the USA are ballot access and the reduction / elimination of voter fraud. Many groups see these as opposing forces as measures designed to reduce fraud can seemingly have the effect of reducing ballot access, and measures designed to improve access can seem to have the effect of making fraud easier.

Inno.vote aims to address both fraud and access simulatneously.

Unsupervised Remote Voting in Estonia

Estonia has had some form of online voting since the 2005 elections in that country. Adoption in Estonia started at 1.9% and has since grown to over 30%. The Internet voting system is built upon the mandatory national ID card scheme in Estonia. These ID cards each contain a smart chip (similar to a cellphone SIM card chip) that contains an private key that is used in those elections encryption schemes.

This system has garnered criticism that it can be corrupted, especially in the tabulation of votes. This is something that the SecureVote standard alleviates through its key handling protocols and ballot broadcast procedures.

Barriers to Adoption in the USA

In the USA, elections are not conducted by a centralized authority, as they are in Estonia, or in much of the rest of the world. As a result, many elections officials make many decisions and set many standards which affect the design and creation of voting systems. Most significant in the USA is the Elections Assistance Commission (EAC) whose Voluntary Voting Systems Guidelines (VVSG) set standards which are adopted to various levels by many (but not all) states.

Regimes like mobile device voting have to set strict protections for voter fraud, but also work to improve ballot access, giving all citizens a voice, and making sure that our elections are secure and the results trustworthy.

Local Government Adoption

Elections are run at the state or local government level, and they are ultimately responsible for the delivery and management of their election lifecycle, including the technology used. These jurisdictions typically outsource parts or all of their elections technology to bidding vendors via

competitive procurements (RFP / RFQ). The technology sought can include any combination of elections hardware, registration, EMS, and voting software.

Inno.Vote intends to leverage its technology and implementation team to respond to these government procurements with our secure and open-source technology and low cost as discriminators from the competition. Jurisdictions typically spend $12-$15 per voter in their elections, while Inno.Vote targets a cost of $3-$4 per voter. This combined with the extra security and transparency that is inherent in the solution will be a compelling option for most jurisdictions. Inno.vote will respond to procurements asking for a full suite solution or a subset.

Government agencies often issue a Request for Information (RFI) where vendors are invited to document their offerings to give the issuer a high level strategy of technical approach. These are an opportunity for Inno.vote to respond in order to educate a local or state government on its technology and offer some new options and flexibility for when the procurement cycle officially begins and an RFP is posted.

As a marketing strategy, Inno.Vote also intends to proactively demonstrate its technology to various jurisdictions across the country in an effort to raise awareness and shape the way future RFP's are written. In an effort to bolster referenceable accounts, Inno.Vote will reach out to various Universities and other non-governmental civic societies such as unions or NGO's to run their elections using our technology. This will greatly impact the success of proposals written as they will be backed by suitable past performance.

Improving Ballot Access

The SecureVote standard improves ballot access at the polling place by making hardware commodity rather than specialized. This has the effect of bringing down the price of voting hardware, as well as using the state of the art assistive technologies that are built into modern computing hardware. At a lower price point, jurisdictions can afford to purchase and deploy more hardware, reducing the length of lines, a common voter complaint. Additionally the commodity nature of the hardware means that less training and specialize support is necessary.

In addition to making elections more cost effective to administer broadly, Inno.vote makes it easier for first-time voters to register, and gives immediate feedback on voter registrations, and other interactions with the registering authority. The mobile-based nature that this solution enables makes early and absentee voting more accessible, and fits civic duty into the voter's schedule, rather than forcing the voter to arrange their schedule around their civic duty.

Enhanced Measures Reducing Voter Fraud

The SecureVote registration process makes defrauding the registration system impracticable both for individuals and state actors. Since registration ties into DMV and other databases in real-time, legitimate voters and individuals can be quickly identified. Additionally, for voters

without these identity credentials, we can capture documents and faces for human-based verification at the elections authority, and then ensure that those same voters are the ones voting in elections. This is the equivalent of having in-person registration for every individual voter with a trusted registration clerk. However, even with all of these increased fraud measures, we will still be able to provide access to disadvantaged voters, or to voters for whom this is their first interaction with a government system.

Inno.vote Services and Products

Inno.vote will facilitate the implementation of the SecureVote software for local elections authorities all over the world, focusing on jurisdictions inside the USA first. Inno.vote will provide software integrations with various state and local databases; assist agencies with HAVA and ADA compliance; manufacture hardware accessories to aid polling place implementations; and provide Elections as a Service (EaaS) providing a full suite of software management for local elections authorities.

Integration & Implementation Services

Local elections authorities have to integrate with a number of different external systems, ranging from the multi-state ERIC database, to their state DMV database, to the Social Security Administration's data services and legacy data systems within the jurisdiction. Inno.vote will provide services to these authorities, creating "integration agents" that will collect data from these external sources and integrate it with the various data stores in the SecureVote system.

Inno.vote will also provide consulting services to help local elections authorities plan and execute their implementations of SecureVote backed solutions. These can include running SecureVote software on systems in their own datacenters, and private or public cloud instances.

Both of these types of services will support and be compatible with the Elections as a Service (EaaS) approach.

Elections as a Service (EaaS)

Inno.vote believes that the most popular method to run SecureVote software will be the Inno.vote EaaS product. This product will deliver a complete suite of SecureVote software, as well as the ability to run any number of first or third party "integration agents," and will be delivered on a cost per registered voter (CPRV) model. This will allow jurisdictions to clearly compare the cost of the Inno.vote EaaS platform with other solutions, as well as the cost of upkeep of their current legacy systems.

The EaaS product will always run the latest confirmed version of SecureVote software, maintaining security sweeps of all running machines, and having machine instances that live no longer than one week, and are always fresh for an election. By cycling instances regularly, if

there is a corruption of or intrusion into any underlying software, the scope of that intrusion will be so limited as to be useless to the potential attacker.

The EaaS product will follow all of the best practices and qualities of highly scaled systems that local authorities can benefit from, but do not have the resources to implement locally.

Training

Inno.vote will provide training resources and classes at all levels from poll worker through to systems administrator for working with SecureVote and Inno.vote software, hardware and services.

In addition to a complete library of training videos, Inno.vote will host regular webcast sessions on how to use the various software, as well as in-person training sessions.

The company will also provide election day on-site support workers to jurisdictions.

Polling Place Solutions

For the polling place, Inno.vote will sell a poll pad set, a ballot booth set, ballot boxes, local tabulator as well as iPads, iPad stands, printers, battery backups and enclosures.

Though the SecureVote standard is built to work on commodity hardware, Inno.vote will provide a complete set of hardware solutions for polling places. This will include complete sets of commodity devices, accessories, and carrying cases, as well as partial solution sets for jurisdictions providing their own devices.

Data Mobility, Integrity and Replication

Data replication and mobility between systems is one of the biggest challenges facing elections authorities. They often deal with many systems with different databases and varying data replication protocols. The SecureVote suite of software aims to make data robust within the system, as well as transportable between system components and easy to export and import.

Data Mobility with Integration Agents

The “integration agent” model that the SecureVote system implements aims to make it such that data can come from or go to any source without having to be aware of SecureVote’s internal message bus (though either integration agents or external providers may listen to and talk on this bus).

Each integration agent will either listen for updates, publish updates, or both. Agents that listen for updates will get updates on Registration records, Contest records, Contestant records, Election records, Worker records.

Definition of Entities & Roles

Elections Authority: The elections authority is any BallotChain address owner that publishes an ElectionStart transaction. For the purposes of this paper, we generally refer to a local elections authority (such as a county board of elections) in the USA, but this can be generalized to any organization running an election, from a corporate shareholder meeting to a school government to a national plebiscite.

Observer: An observer is any group looking to observe the operation of an election and confirm the result of the same. This could be a democracy focused non-government organization (NGO) such as the National Democratic Institute, a government entity such as the Elections Assistance Commission or a not for profit such as the Pew Charitable Trusts.

Voter: A voter is any individual who is authorized to participate in an election.

Precinct Captain: The precinct captain is the individual in charge of the polling place

Check-in / Ballot Clerk: The Check-in and ballot clerk is the individual responsible for creating ballot tokens on behalf of voters at a polling place.

Central Elections Manager: The central elections manager is the individual responsible for generating the election events via the Election Management System.

Poll Pad: A poll pad is software used to check in voters, ensure that they have not already voted and to generate their ballot token at the polling place.

Precinct Tabulator: a precinct tabulator keeps a tally of all votes cast within a precinct and will also act as a miner in the private and protected BallotChain networks.

Central Tabulator: a central tabulator processes a complete BallotChain and acts as a mining node on a BallotChain network.

Voting Client: A voting client can be a smartphone app installed on a voter's personal mobile device, or can be in a polling station. When on a voter's device it will also contain a poll pad type functionality in addition to the voting client itself.

Registration Database: the SecureVote registration database contains all registration and enrollment information for all voters in the system.

Legacy Registration System: a legacy registration database contains registration but not enrollment information for voters. This information is to be used as the basis for enrolling voters in the SecureVote system.

Election Procedures

Voter Registration Procedure

Voter registration is used to ensure that only authorized voters are participating in an election. This is similar to the authorization, rather than authentication phase of a sign-in. Our registration process consists of two phases, registration and enrollment. Registration is the collection of changeable voter data (e.g., registration address or party affiliation) and the verification of government-verifiable data (e.g., name, SSN, DLN, DOB) with external databases. The enrollment phase is the collection of additional data that can be used to later verify that the authorized voter is actually the same individual who has enrolled in the process. This includes the collection of biometric and security data such as a fingerprint, facial recognition scan and passcode setting. Diagrammatic discussions of these processes can be found in the "Registering the individual to vote" section above.

Elections Setup Procedure

Elections are to be set up prior to their start. The setup phase includes the definition of all contests, the definition of geographical precincts, the ballot styles which are to be presented to voters residing in those precincts, as well as all other associated metadata for the election itself.

Election Start Procedure

To start an election, the elections authority must first generate a public/private keypair. The private key should be printed and stored in multiple safe locations (as it will not be saved at this time). The public key is then transferred to the EMS where it can be used to generate an ElectionStart transaction.

Election End Procedure

To end the election, the administrator must take the previously printed private key and enter it into the EMS (this can be done via QR code, as these codes can encode 2kb of information each). The private key is then validated against the public key which started the election, and, if it matches, is published in an ElectionEnd transaction.

Important Concepts

Encryption Schemes Used

In this paper we will mention many types of encryption. In general, when we talk about key pairs, public keys, private keys or signatures with private key, we are discussing operations using the **RSA Encryption Algorithm with a 4096-bit key (a.k.a. RSA-4096)**. We will also discuss encrypting something with another actor's published public key. In almost all instances, this involves generating a random secret that is smaller than our message, encrypting the secret with the recipient's public key and then encrypting the message with a symmetric cipher, in our case, **20-round AES-256**. Since the ballots themselves will be actually encrypted using a symmetric cypher, it is worth noting that an attacker could decrypt ballots prior to the end of an election, but current technology does not allow this (i.e., it would take many billions of years to decrypt a single ballot by brute force. It is possible that AES would be "broken" in the future, it is unlikely that an attack currently exists or will for the foreseeable future. It is worth noting further that this symmetric cipher could be easily swapped for another algorithm.

Blockchain Technology

A blockchain is a peer-to-peer distributed database that verifies each additional transaction by hashing it with the transactions that have preceded it. Blockchains were originated by Bitcoin and are a generalized distributed ledger system. More information can be found on the Bitcoin wiki (https://en.bitcoin.it/wiki/Block_chain).

Hash IDs

A number of items in this scheme use **SHA256** hashes. These IDs are used to identify contests, contestants, and machines in the system. It is important to note that each of these is a SHA256 hash computed on 4kb of data read from the generating device's entropy source concatenated with an **RFC 4122 version 1 UUID**.

Artifacts of the System

The BallotChain

The BallotChain is a blockchain-based altchain where transactions are one of the system artifacts listed below, an ElectionEvent, a BallotToken, a VoteReceipt, or a SecureVote. All elections have three BallotChains associated, a private, a protected and a public. Private and protected BallotChains have a restricted peer list and require permissions to join the network, the public BallotChain lacks this restriction.

All nodes in the BallotChain network will contain a full copy of the database, and any node can verify that all transactions are valid and from there recreate the full context of an election.

The ElectionEvent

An ElectionEvent can be one of five types of transactions, the ElectionStart, the ElectionEnd, the BallotStyle, the ElectionVoterJoin and the BallotTrump.

The ElectionStart Transaction

The ElectionStart event is the beginning of a new election. This event contains the following fields: election name, election id, election start datetime, election end datetime, election publickey, and election contests with contestants. This gives us the particulars of an election, enough to validate any ballot style we are given as containing valid contests and contestants, as well as the public key used to encrypt all ballots in the election.

The ElectionEnd Transaction

The ElectionEnd transaction signifies the end of an election. Only transactions taking place after the ElectionStart and before the ElectionEnd transaction can be considered valid parts of the election. The ElectionEnd transaction contains the election id to match the ElectionStart transaction, and the private key that matches the public key published at the ElectionStart.

The BallotStyle Transaction

The BallotStyle transaction publishes a ballot style to the BallotChain. The BallotStyle provides information on which contests are to be voted on, in which order these contests should appear, as well as any styling or typographical data associated with the contests or with the ballot. A BallotStyle transaction includes the BallotStyle itself, as well as the ElectionId for which the ballot is valid.

The BallotTrump Transaction

A BallotTrump transaction can be issued only by an elections authority (the address that published the elections key) during an election, and invalidates a previously redeemed BallotToken. The transaction contains an ElectionId as well as an ElectionVoterJoin address. Any BallotToken issued to the ElectionVoterJoin address prior to the BallotTrump is invalidated, whereas the next one after is considered valid.

The ElectionVoterJoin Transaction

This transaction is issued by the elections authority when a voter joins an election. This is the equivalent of signing the poll book. The elections authority will maintain a private database of which registered voters have been issued ElectionVoterJoin transactions (and the associated addresses), but not which address or transaction was issued to which voter. This information is kept secret by the voter (or her voting client) and is used to obtain a BallotToken. The ElectionVoterJoin transaction contains the BallotChain address of the voter, the BallotStyle that that voter is entitled to vote and the Election's Id.

The BallotToken

The BallotToken includes a BallotStyle id (referencing a previously published BallotStyle), an ElectionId, the ElectionVoterJoin address to which the token is issued and the token itself (another address on the BallotChain). The BallotToken is then referenced by any SecureVote cast on that ballot. This is to ensure that any SecureVote cast is being cast by an authorized voter.

The VoteReceipt

The VoteReceipt is an artiact issued to a voter directly that contains a secret value that is encrypted in their ballot for the use of authorized tabulators only. The voter can then go and query any authorized tabulator with their VoteReceipt and see if the ballot that they cast is included in that tabulator's verified vote total. The tabulator public keys are made available via the BallotChain, but are never to be released and are to be destroyed following an election. The secret in the VoteReceipt is separate from the VoterElectionJoin address, the BallotToken, or anything else linked to a voter. This is linked to the ballot itself, and only the voter herself is ever made aware of the secret. It is not externally recorded anywhere.

The SecureVote

The SecureVote transaction is the ballot cast by redeeming a BallotToken. The ballot is encrypted with the Elections public key and cannot be decrypted until after the election has concluded. The data contained in the SecureVote envelope is discussed below.

SecureVote Envelope Definition

A SecureVote envelope is the packet of information that completely describes a cast ballot. This envelope is then encrypted using the election's public key. SecureVotes are then recorded to multiple blockchain networks, one comprised of public observer nodes, a second of trusted observer nodes, and the third of a second group of trusted observer nodes. This allows for elections in case of a disagreement. Following the conclusion of an election, the election private key is published, allowing SecureVotes to be decrypted and independently verified. The SecureVote envelope consists of two parts, the ballot itself, and a signature block. An example unencrypted SecureVote can be found in Appendix C.

Ballot Block of SecureVote Envelope

The sample ballot contains eight major components, the authority, the ballot style, the ballot and the receipts. The authority is the domain of the authority administering the election, the ballot style represents the list of contests displayed to the voter, the ballot represents the choices made by the voter, the election an ID for the election, the location the ballot was cast, the receipts are encrypted with the public key of the recipient tabulator, the timestamp, and the version of the API under which the ballot was cast.

Authority

The domain of the authority under whose aegis the election is held. The authority publishes all information related to this election on the domain listed in this field. This field also serves as an API endpoint, serving the version of the API listed in the ballot. The authority will publish information about all elections it mediates.

Ballot Style

The ballot style is represented by a Hash ID, and all ballot styles valid in an election are published by the named authority. The ballot style maps back onto a style published by the authority. A style consists of a listing of contests, where each contest is represented by its random SHA256 hash (as opposed to an ordinal ID number), as well as styling and other meta-data as necessary.

The API will serve a ballot style when the style is looked up by its Hash ID.

Ballot

The ballot itself is constituted of two parts, *contests* and *choices*. Both of these fields are arrays consisting of a list of items, where the caster's choice for *contest[n]* is represented by *choices[n]*. Each contest is represented by its random SHA256 hash. Information about the contest can be looked up from the authority using this hash. Each choice is also represented by a hash. The elections authority will publish information about all contestants that are legally on the ballot or

write-ins that receive more than 0.01% of all votes cast. In this way the authority may protect against coercion attempts that make use of the write-in field to encode information.

Contests

Each office or question being contested will receive a Hash ID. The API will provide a listing of all contests in an election, as well as the contestants valid in each contest, as well as metadata surrounding the contest. This metadata will contain the contest format (i.e., winner take all, instant runoff, ranked preference voting, etc) and any other information relevant to replicating that contest on a ballot.

Candidates / Contestants

Using the API, a contestant may be looked up by hash or by name. Similarly, a list of all contestants receiving votes in a contest may be retrieved using the contest Hash ID. When looking up by Hash ID or by name, all valid Hash IDs for that contestant will be returned. Only "published" contestants can be looked up by name (see above on write-in contestants for anti-coercion measures related to this).

Write-in Candidates / Contestants

When a write-in contestant is chosen for a contest, if the machine recording the ballot has not seen this contestant before, it will generate a Hash ID for the candidate (format discussed above), and will record it immediately, as well as sending it to the tabulator(s) with which it is registered. If the machine has already encountered this write-in value, it will use the same previously recorded value. The tabulator will combine all votes recorded for each write-in and generate a new Hash ID. The candidate name, new hash ID, number of votes and all unique hashes will be sent to the central tabulator(s), encrypted with the central tabulator's public key. The central tabulator will perform the same task of combining votes, and if a candidate garners more votes than the coercion threshold set by the authority (by default, 0.01% of all votes cast), then all of the hashes corresponding to that contestant will be published.

Election

This is the HashID for the election itself. The elections authority will publish, via API, a list of all valid election HashIDs, and the API will provide information around the contests in an election, ballot styles in the election, and contestants in each contest.

Location

The location is a normal latitude and longitude of where the machine recording the ballot was when the ballot was cast.

Receipts

The receipts block of the ballot contains a number of recipients and receipts. The recipient is the HashID of the tabulator for which the receipt was encrypted. The cyphertext in the "receipt" field

of each of these contains the same plaintext receipt number (also a HashID) given to the voter on their voting receipt. When a tabulator decrypts and tabulates a ballot, it will take the receipt that it can decrypt, decrypt it, and publish it **separately** from the vote. In this way, we can ensure that each individual vote counts, as well as getting a listing of all of the votes (by receipt ID) that were counted in each contest. The private key exists only on the tabulator, is generated at the beginning of an election, and destroyed after the results are certified. Once the private keys able to decrypt the receipts are destroyed, it will become impossible to trace a vote back to an individual voter, no matter how much other information is available.

The ideal case here is that there is a tabulator inside the voting precinct (the “tabulator” machine), as well as at least two others tabulating results from the precinct. This would also introduce an idea of a “trusted observer” who would be able to run an external tabulator to externally corroborate the result of an election. Voting machines are instructed which tabulators to encrypt for when they “join” the election, and then throughout the election as tabulators might join or drop off.

Timestamp

This is a common unix timestamp. All ballot marking devices will sync with network NTP devices. As ballots are published onto multiple blockchains, this is not an important “source of truth”, but rather represents the time at which the ballot marking device believes the ballot was cast.

Version

This is the API version under which the ballot was cast. It’s a 4-number dotted string in the form of *MM.mm.pp.bbbbbb* where *MM* is the major version number, *mm* the minor version number, *pp* the patch level and *bbbbbb* is the build number (unique across all versions).

Signature Block of SecureVote Envelope

The signature block of the SecureVote envelope contains a “signer” which is the HashID of the machine signing the SecureVote, and the signature itself. The signature is an RSA signature of the Ballot block of the SecureVote envelope computed using the private key of the signing machine, and verifiable using the public key of that machine.

Ballot Style Definition Format

About Inno.vote

Inno.vote is organized as a Delaware corporation. The purpose of Inno.vote is to sell implementations, services and software to state and local jurisdictions to run elections on the

SecureVote platform. Its primary products are consulting services, the creation of integration software between state and local legacy systems and the SecureVote open source suite of software, and it also provides EaaS, Elections as a Service.

About the Security and Transparency in Elections Foundation

The Security and Transparency in Elections Foundation is a not for profit corporation whose mission is to manage and further the SecureVote suite of open source software. Inno.vote is a primary contributor to the foundation and is allocated two seats on the five-member board. The other three members are nominated and elected by contributors to the software, with each active contributor having one vote. An active contributor is defined as an individual github profile who has contributed one commit to a pull request, or opened one issue in the issue queue on github.

Appendix A — API Reference

The Voter Registration System

The Elections Management System

The Pollbook System

The Tabulator System

Appendix B — Security Protocols

Casting a Ballot

Let the elections public key *E* and tabulator public keys *T1, T2, T3*

Let the ballot *B* and the ballot receipt secret *R*

Let the public BallotChain *BC(c)*, the protected BallotChain *BC(r),* and private BallotChain *BC(p)*

Steps:

- *R* is encrypted using *T1* resulting in *R(e)1*
- *R* is encrypted using *T2* resulting in *R(e)2*
- *R* is encrpyted using *T3* resulting in *R(e)3*

- *R(e)1, R(e)2* and *R(e)3* are added to *B*
- *B* is encrpyted with *E* resulting in *B(e)*
- *B(e)* is published to *BC(p)* then *BC(r)* then *BC(c)*
- **IFF** all previous steps successful, voter is supplied with *R*

Registering a New Device

Appendix C — Code Samples

Example SecureVote Envelope (Unencrypted)

```
--------------------------- BEGIN SECUREVOTE ---------------------------

----------------------------- BEGIN BALLOT -----------------------------
{
        "authority" : "secureelection.kingcounty.gov",
        "ballot_style" : "2696e2d24fa5a8264065a2bcf6fb143f1f211c4bea3281d5983bdd6766e4f453",
        "ballot": {
                "contests": [
                        "25555d6cd6ad31cfe7b1c00b27c80b573d519d2e562aebd63312fe1c12f47e8d",
                        "3359190149cdd72ffe1efcb46c6dfcbce6aebdc70bb93d7cd896c8c9eaaf02ff",
                        "50785592ae6344bc25bc34ca88533fa174a7fb6090e65dadfc96b806bb069f66",
                        "0eb91bdd8c06ea2cdf9c0f8b7df408f289b9ec65ce814cc172c97c66a7586b9d",
                        "d98d8db304948dcfe40943b9ebe622bea02f8269b7e4f3203f6869cc269bcf3d"
                ],
                "choices": [
                        "3f874577b5f36c227a3a35e72b15f9415da7cadc2ea6e6b408a6f5c8126c1c96",
                        null,
                        "2b718cd0547b90e6ddc43a9dee371ca520978cc195338b3f039423775f294a95",
                        [

"155459ae1da4a5070c4819ea229a50f308c4959a0ec332045a26589342852d1e",

"8c9e3d8044bae8898c29ba27f6cb9850eb710f005c9eefb2e0bc0daceb4cd7d9"
                        ]
                        "a9d36fe69f6fd2ced17ae9033088406074430bc39c49bee961c57e283b5f27d8"
                ]
        },
        "election": "23cd00b83d5eec836cff5c452eaf4b092acdaad293bf4f8f77df2b223f8130ca",
        "location": [39.00000,-77.00000],
        "receipts": [
                {
                        "recipient":
"1eafbab1c0e74be2ffea9cdfaa288e3b62ae7eef6d7d37776bbaf1659001ef2b",
                        "receipt":
"d94d889e88853dd89769a18015a0a2e6bf82bf356fe14f251fb4f5e2df0d9f9a94a68a30c428b39e
3362fb3779a497eceaea37100f264d7fb9fb1a97fbf621133de55fdcb9b1ad0d7a31b379216d7925
2f5c527b9bc63d83d4ecf4d1d45cbf843e8474babc655e9bb6799cba77a47eafa838296474afc24b
eb9c825b73ebf549"
                },
                {
```

```
                        "recipient":
"4ec7b34b3ee4ee7654221761ead48446c296bafde95358c92f76b5fc9e96131a",
                        "receipt":
"47b9cfde843176b88741d68cf096952e950813151058ce46f2b048791a26e507a1095793c12bae1e
09d82213ad9326928cf7c2350acb19c98f19d32d577d666cd7bb8b2b5ba629d25ccf72a5ceb8a8da
038906c84dcdb1fe677dffb2c029fd8926318eede1b58272af22bda5c5232be066839398e42f5352
df58848adad11a1"
                },
                {
                        "recipient":
"b3006353cf3e990d7d0bbb8eec0533af03c17e18af84ad66ef64c9526bee5836",
                        "receipt":
"53537090f9ee7af5443be6450e2482418f1e637e0ceaff6030d32ac23df2c78ff1c3b73530b9e081
f23a37b7ea5270d6bd28dc16a2bb548985e6af93c31d61f56f87c4491dd9833dfd2c73243a4a674e
a88b370c476ff4acfef23aaa2a87fcc615dd8cd3bcfe0de2b83b23469e4206aa33b1fd64a1ee95c3
f598401082955da9"
                }
        ],
        "timestamp": 1460400189
}
------------------------------ END BALLOT ------------------------------

------------------------ BEGIN BALLOT SIGNATURE ------------------------
{
        "signer": "cb3edf2b9c1b2cc44638e307ab42c9b5656e85cbbd25c202acb6cbcdfce960c1",
        "signature":
"4cf4e6530eb015fe0e7701967f3dc5fe15d59b506f0bb5efb8c0fa22df43f86c9cd1e1e8f9e4e3e98f7436584e7e9
84c9b2c40092d213716287494671602b93e119b24c3d8efa354d2a080357a5e44d0a71755b0e96f295fc490120a5f7
c311c7353905ac573e0c7cf245390f8617cdaf9611758eb6eccbf340ffb32d8a43e83"
}
------------------------ END BALLOT SIGNATURE ------------------------

---------------------------- END SECUREVOTE ----------------------------
```