Business Considerations

1. The number of voter registration organizations (VRO) that will join the system
2. The number of states that are willing to join the system and their respective time frames to do so
3. The level of bespoke customizations may be an issue, but at first glance that does not seem to be a big issue. However, there will be some customization and changes needed at each state's system, which will involve collaborating with the state system support vendor.

Political Considerations

1. What will be the key driver for the states to join the system? - most states want increased voter participation and outreach, so the VR groups basically become an effective and inexpensive extension of the states and counties.
2. Alternatively, is there a key resistance factor that may make states disinterested or disinclined to join? - this does not seem to be the case.

Resource Considerations:

1. Technical: System arch, design, security, development, test and implement
2. Program Management: Overall mgmt of and coordination across various stakeholders. Liaison with the state agencies.

Roles and Responsibilities:

1. Technical: Architecture, security and development of the solution
2. Program Management: Overall management of the VRO's and EA'S
   a. Work with the voter registration organizations to establish state candidates probably with a phased approach to joining the system.
   b. Work with the states for information/requirements gathering, specification definition, data exchange/interface protocols, coordinating testing and implementation
   c. Work with the Voter Reg groups to establish clear exchange protocols, install encryption program, coordinate data exchanges and testing.
   d. Work with the technical team at Galois to establish clear system specifications, exchange protocols, data specifications and bespoke state system requirements.

Cost Considerations

1. Program Management
2. Technical - Galois
3. EA specific interface customizations

The Proposed Solution
Timeline:

Requirements definition and system spec	: Week 1-4
Database design and exchange protocols	: Week 1-8
Security and multi-platform encryption architecture	: Week 3-10
Network and system architecture	: Week 4-8
System development and testing	: Week 6-17
Functional and state testing	: Week 18-20
Feedback incorporation and finalization	: Week 19-21

Cost:
The cost of the technical piece of this proposal that includes system development and internal project management of the Galois development project is $91,000. Our effort analysis of this architecture indicates approximately four-to-five person-months of work is necessary to design and build a high-assurance, open source product that fulfills these requirements. The cost of such work for us with no margin or room for error is $63,000, thus we propose to price this work at $91,000 to accommodate for changes in requirements, errors in effort estimates, etc. In addition to this, there is about $250/month or $3000/year cost of hosting/deployment in the cloud that is ongoing.

Exclusions:
This estimate does not include additional resources that are necessary for project success, including any of the program management activities stated earlier, In addition, coordination with the EA for information gathering, data exchange and testing coordination or implementation activities at each EA, partner VRO integration work etc.

Prerequisites:
Information gathering and EA specific requirements need to be gathered early enough to ensure design does not progress without key requirements.

Dependencies and Constraints:
1. State specific interfaces whether batch vs. flow
2. Deployment of security decryption modules on state (EAs) systems
3. Exchange frequency and protocols to be negotiated with participating EA
4. Total number of EA's that are going to be part of this project in phase I
5. Implementation of each EA interface completely depends on state schedule and availability for testing
6. Archiving needs - not significant since the dataset is not large
7. Images of and VR cards/applications that may need to be exchanged.

Solution Highlights:
1. Cloud based high assurance open source software with end-to-end encryption of all data transferred over encrypted connections
2. End-to-end encryption of all data, no decryption of data at any point in time to ensure security, with no risk of personal data leakage even in case of hacks
3. Standard API for VR groups to upload data to the system

4. Unified data model to cater to the needs of all the state VR systems
5. Low cost of data storage and maintenance
6. Easy and quick addition of subsequent VR groups, and relatively fast addition of Electoral Authorities (EAs)

Technical Solution

The OVR implementation we propose has three components: a library used by voter registration organizations (VROs) to securely upload voter registration data; a library used by electoral authorities (EAs) to securely download voter registration data; and a collection of data storage servers (DSSs). All communication among these system components is carried out over encrypted Internet connections. A key aspect of our implementation is that all voter registration data is encrypted *end-to-end*: each record is encrypted before a VRO sends it over the Internet to a DSS and remains encrypted until it is received by an EA. Voter registration data is *never* decrypted within the system, even while being processed by a DSS for delivery to an EA. This allows DSSs to be hosted on an inexpensive public cloud infrastructure such as Amazon EC2 or Microsoft Azure, with no risk of personal data leakage even if the cloud infrastructure or any number of DSSs is successfully hacked.

The library used to upload registration data will provide a minimal, easy-to-use API interface to multiple programming languages (including at least C, Ruby, Python and PHP), for seamless integration into existing VRO systems. The library used to download registration data will produce data files in formats appropriate to each particular EA, for easy integration into existing individual or batch voter registration processes. Both libraries will also be available as standalone applications, if appropriate for a specific VRO or EA installation. Each DSS will use standard, open-source database technology augmented with the capability of performing matching operations on encrypted records; this will allow it to determine the appropriate voter registration requests to send to each EA while leaving the data itself encrypted at all times. We anticipate that the volume of data to be handled by the system will necessitate only a single DSS, but will support the use of multiple DSSs to guarantee system availability and data preservation in the event of hardware failures.

We intend to not hand-implement the interfaces for each VRO and DSS. Instead, we will define a Voter Registration Domain Specific Language (DSL)---which we will call the Voter Registration Language, or VRL for short---that will permit us to describe the VRO API and each EA's voter registration data model in a handful of lines of code, then automatically and correctly generate all VRO APIs and EA data models. This means that adding a new feature to the OVR or a new EA will be as simple as writing a short description and pushing a button—not writing any new code.

The technical tasks associated with this project are:

1. import/define the voter registration data model that unifies all voter registration data models across all EAs

2. define the VRO API and cryptographic protocol, and choose which client languages we will support
3. define the EA cryptographic protocol
4. define the VRL and design and develop the VRL compiler
5. use the VRL compiler to generate VRO APIs and EA data models for a representative sample of VROs and EAs
6. design and develop the OVR DSS

We believe that we can do a demonstration Amazon cloud deployment during the project execution for free, as their smallest EC2 instances are sufficient for test deployments. Our analysis indicates that deploying the OVR system to the entire nation on Amazon EC2 with a 90 day data retention results in a deployment cost of approximately $250/month ($3000/year) given our architecture.