InnoVote Systems Database

Detailed Design

By
Erin Thead
Software Engineer
erin@erinthead.com

Table of Contents – Database Detailed Design

1. Introduction

1.1. Purpose.

The purpose of this document is to communicate a suggested design for the databases that the InnoVote election products will use. The document provides a description of the databases' structure, rules, data protection, and user authentication.

The intended audience of this document is the developer and any other persons interested in the project, including election reform activists, computer security professionals, political figures with an interest in election reform, and potential buyers of the design.

1.2. Scope.

The InnoVote line of election products will need to store election data in databases. In deployment, every InnoVote system will have a local database and DBMS. Since the integrity of the election data is of paramount importance, these databases must be designed to protect sensitive data. Existing election databases allow easy modification of sensitive data and rely on an "honor system" for election officials and election software vendors.

As is described in references [2], [6], [7], and [8], numerous software functions of InnoVote products require a secure database to operate correctly. The design proposed in this document will provide the required security to protect the integrity of sensitive election data processed by InnoVote systems.

1.3. References.

[1] Thead, E. *InnoVote CardReader Hardware Requirements Overview*, 2005.
[2] Thead, E. *InnoVote CardReader Functional Design*, 2005.
[3] Thead, E. *InnoVote Database Access Matrix*, 2005.
[4] Thead, E. *InnoVote MyVotronic Hardware and Operating System Overview*, 2005.
[5] Thead, E. *InnoVote Network Detailed Design*, 2005.
[6] Thead, E. *InnoVote ReliaVote Central Server Functional Design*, 2005.
[7] Thead, E. *InnoVote ReliaVote Precinct Edition Functional Design*, 2005.
[8] Thead, E. *InnoVote SecureDRE Functional Design*, 2005.
[9] Thead, E. *Security Analysis of InnoVote Products*, 2005.

1.4. Overview.

The remainder of this document is organized in the following fashion:

Section 2: Contains an overview of the system, including a diagram of the relationship of the Database and database management system to other components, and a graphical depiction of the organization of the Database in the form of an entity-relationship diagram (ERD).

Section 3: Provides a detailed description of the purpose and all attributes of the entity and relationship sets (or “tables”) that will be present in the Database, as well as restrictions on the data that are permitted in each table.

Section 4: Provides a detailed description of cryptographic measures to ensure the secrecy and integrity of sensitive data stored in tables of votes.

Section 5: Provides a detailed description of users that the Database will recognize and the permissions associated with each user.

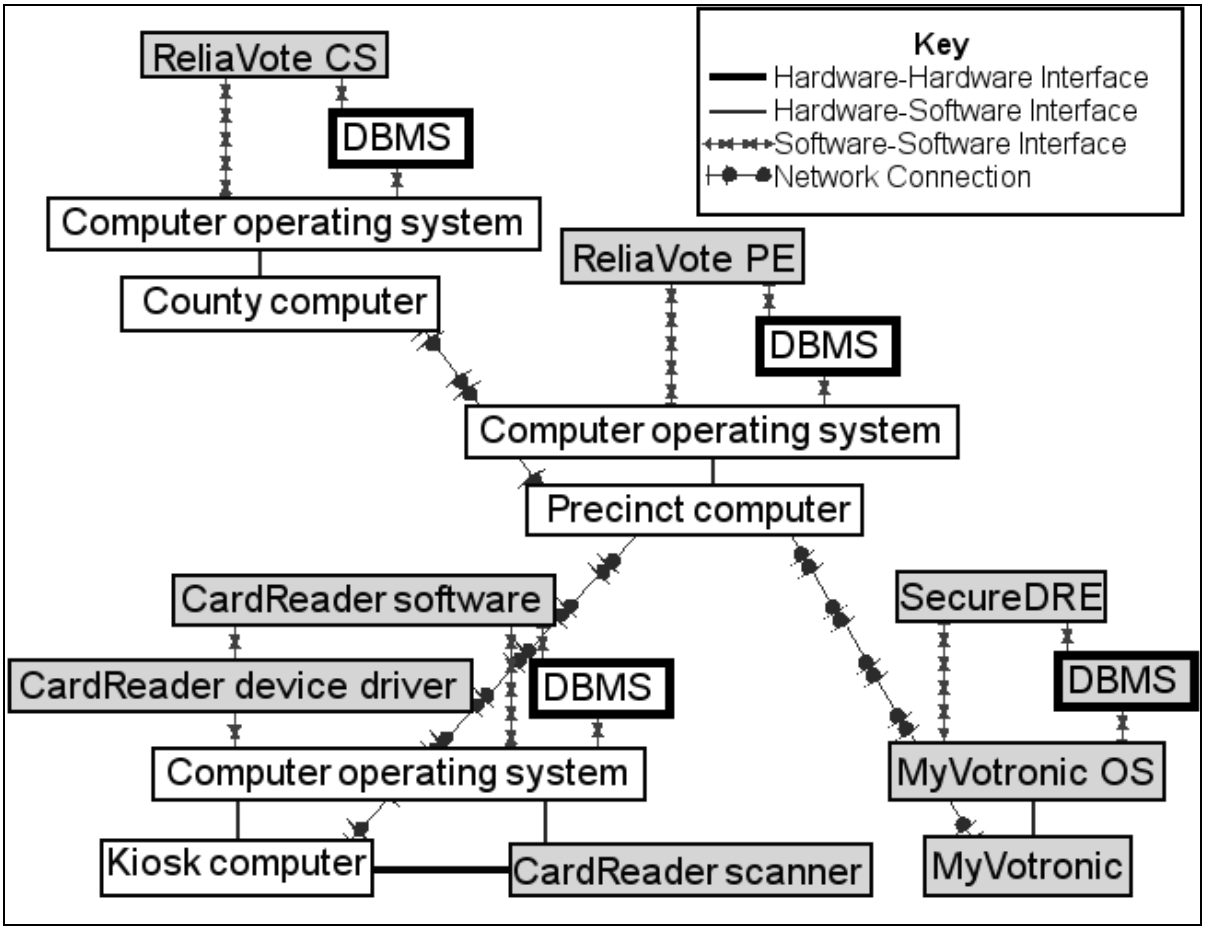
2. Overview

2.1. Deployment of the Database.

Figure 1 shows the deployment diagram for all InnoVote products and necessary third-party components. Light gray boxes represent InnoVote products; white boxes are non-InnoVote systems that InnoVote products will use. Items that this document describes are surrounded with thick boxes.

The Database and database management system will be deployed on every system with an InnoVote software product installed. It should be noted that the DBMS can be a standard third-party product for all systems except the MyVotronic. For that product, the DBMS must be able to be installed on MyVotronic OS [ref. 4]. This may necessitate a custom DBMS that is compatible with the third-party DBMS.

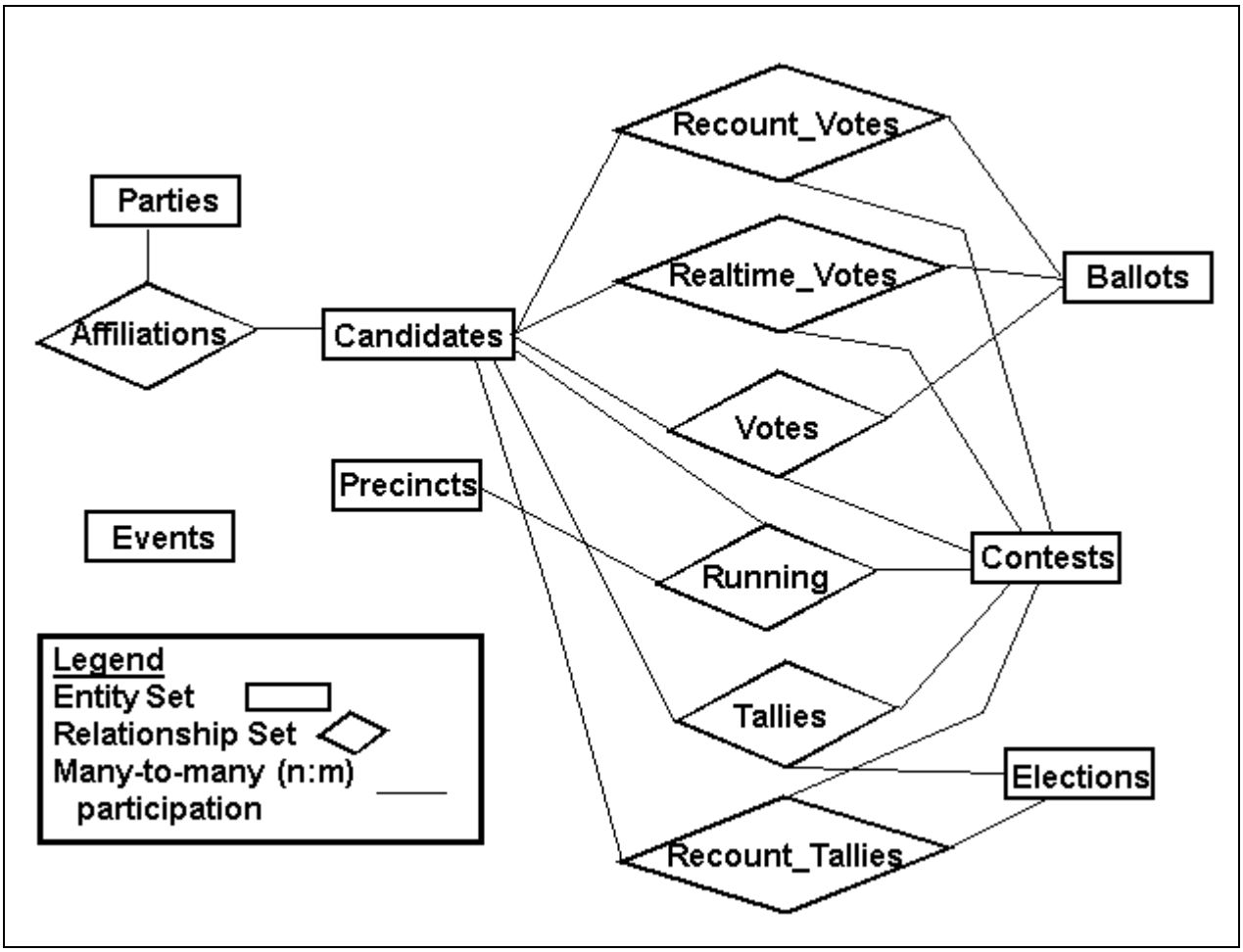
Figure 1: Deployment Diagram.



2.2. Entity-Relationship Diagram.

Figure 2 is an entity-relationship diagram for the InnoVote Systems Database schema. It does not contain attributes of entity and relationship sets, but these attributes are detailed in §3 of this document.

Figure 2: Entity-Relationship Diagram



3. Relational Schema

3.1. Entity Sets.

This section describes the entity sets for the InnoVote Systems Database.

3.1.1. Elections

This table contains a list of all elections for which data are stored in the database.

3.1.1.1. Schema

Elections (election_date, poll_closing_time, random_seed)

3.1.1.2. Description of attributes

Election_date: Primary key in date format. Indicates the date of an election.

Poll_closing_time: Attribute in time format. Indicates the time at which the polls will close.

Random_seed: The random number used by ReliaVote Central Server to generate bar codes for paper ballots. Only one number can be associated with a ballot set for a particular election.

3.1.2. Ballots

This table contains a list of all ballots that are cast in an election.

3.1.2.1. Schema

Ballots (ballot_id, archive_flag, locked_flag, recounted_flag)

3.1.2.2. Description of attributes

Ballot_id: Primary key in string format. Generated from bar code printed on each ballot or by SecureDRE.

Archive_flag: Boolean attribute indicating archive status of ballot. Set by SecureDRE when a vote is modified after a ballot has already been printed.

Locked_flag: Boolean attribute indicating lock status of ballot. Set by CardReader or SecureDRE software when a vote is finalized.

Recounted_flag: Boolean attribute indicating recount status of ballot. Set by CardReader when the ballot is scanned as part of a recount.

3.1.3. Candidates

This table containst a list of all candidates that are running for office in an election. *Alternatively*, it can contain a list of all options that are on a ballot for a ballot initiative. For some races, the option "write-in" will be a candidate name.

3.1.3.1. Schema

Candidates (candidate_id, candidate_name, candidate_info)

3.1.3.2. Description of attributes

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option, chosen by the user when programming the Database.

Candidate_name: String value indicating the name of the candidate.

Candidate_info: String value containing space for any information about the candidate (other than party affiliation) that an election official chooses to add.

3.1.4. Parties

This table contains a list of all political parties that are registered for a particular area. "Independent" can be a political party, as can be "none."

3.1.4.1. Schema

Parties (party_id, party_name)

3.1.4.2. Description of attributes

Party_id: Primary key in integer format. Unique identification for a political party, chosen by the user when programming the Database.

Party_name: String value indicating the name of the political party.

3.1.5. Contests

This table contains a list of all races and ballot initiatives that will be part of an election.

3.1.5.1. Schema

Contests (contest_name, contest_information)

3.1.5.2. Description of attributes

Contest_name: Primary key in string format. Indicates the name of a contest or ballot initiative that will be voted on.

Contest_information: String value indicating any information about a contest or ballot initiative that the election official chooses to add.

3.1.6. Precincts

This table contains a list of all precincts in a particular county.

3.1.6.1. Schema

Precincts (precinct_id, ip_addr)

3.1.6.2. Description of attributes

Precinct_id: Primary key in string format. Unique value identifying a precinct.
Ip_addr: String value indicating the stored IP address of the computer for a precinct.

3.1.7. Events

This table contains a list of all significant events and *all* errors that have occurred over the course of operation.

3.1.7.1. Schema

Events (event_id, event_code, is_error, event_source, event_date, event_time, event_class, event_message)

3.1.7.2. Description of attributes

Event_id: Primary key in string format. Generated by an InnoVote product when an event occurs. *Note*: The programming of the products will prevent multiple machines from generating the same event_id. SecureDRE, ReliaVote PE, ReliaVote CS, and CardReader all begin their event_id with different characters.
Event_code: Integer value indicating the system event code.
Is_error: Boolean value indicating whether the event is an error.
Event_source: String value indicating the machine on which an event occurred.
Event_date: Date value indicating the day on which an event occurred.
Event_time: Time value indicating the time at which an event occurred.
Event_class: Optional string value indicating the "type" of event, if provided by the system.
Event_message: Optional string value for a system event message, if provided.

3.2. Relationship Sets.

This section describes the relationship sets for the entity sets in the Database.

3.2.1. Votes

This table contains the choices of a voter who has cast a ballot.

3.2.1.1. Schema

Votes (*ballot_id*, *candidate_id*, *contest_name*, writein_name)

3.2.1.2. Description of attributes

Ballot_id: Primary key in integer format. Unique identification for a ballot. Associated with the "ballot_id" attribute of Ballots (§3.1.2).

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Contest_name: Primary key in string format. Unique name for a contest or ballot initiative. Associated with the "contest_name" attribute of Contests (§3.1.5).

Writein_name: Attribute in string format. Stores the user-provided name of a write-in candidate.

3.2.2. Running

This table contains an association between the candidates and the contests. This is in case one candidate is running for multiple offices.

3.2.2.1. Schema

Running (*candidate_id*, *contest_name*, *precinct_id*)

3.2.2.2. Description of attributes

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Contest_name: Primary key in string format. Unique name for a contest or ballot initiative. Associated with the "contest_name" attribute of Contests (§3.1.5).

Precinct_id: Primary key in string format. Unique name for a precinct. Associated with the "precinct_id" attribute of Precincts (§3.1.6).

3.2.3. Affiliations

This table contains the party affiliations of candidates.

3.2.3.1. Schema

Affiliations (*candidate_id*, *party_id*)

3.2.3.2. Description of attributes

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Party_id: Primary key in integer format. Unique identification for a political party. Associated with the "party_id" attribute of Parties (§3.1.4).

3.2.4. Tallies

This table contains tallies for each candidate for each contest in a particular election.

3.2.4.1. Schema

Tallies (*candidate_id*, *contest_name*, *election_date*, vote_count)

3.2.4.2. Description of attributes

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Contest_name: Primary key in string format. Unique name for a contest or ballot initiative. Associated with the "contest_name" attribute of Contests (§3.1.5).

Election_date: Primary key in date format. Unique date for an election. Associated with the "election_date" attribute of Elections (§3.1.1).

3.2.5. Realtime_Votes

This table contains a list of all votes that are being created and stored while an election is in progress. It is generated on the actual DRE or ballot-scanning machines. This table exists because the Votes table is not transmitted until an election is over. Its existence allows for some fraud- and error-checking mechanisms.

3.2.5.1. Schema

Realtime_Votes (*ballot_id*, *candidate_id*, *contest_name*, writein_name)

3.2.5.2. Description of attributes

Ballot_id: Primary key in integer format. Unique identification for a ballot. Associated with the "ballot_id" attribute of Ballots (§3.1.2).

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Contest_name: Primary key in string format. Unique name for a contest or ballot initiative. Associated with the "contest_name" attribute of Contests (§3.1.5).

Writein_name: Attribute in string format. Stores the user-provided name of a write-in candidate.

3.2.6. Recount_Votes

This table contains a list of all votes that are counted during a recount. It exists because a local office may have some use for the original Votes table in the event of suspected election fraud, and the original table would need to be intact and unmodified.

3.2.6.1. Schema

Recount_Votes (*ballot_id*, *candidate_id*, *contest_name*, writein_name)

3.2.6.2. Description of attributes

Ballot_id: Primary key in integer format. Unique identification for a ballot. Associated with the "ballot_id" attribute of Ballots (§3.1.2).

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Contest_name: Primary key in string format. Unique name for a contest or ballot initiative. Associated with the "contest_name" attribute of Contests (§3.1.5).

Writein_name: Attribute in string format. Stores the user-provided name of a write-in candidate.

3.2.7. Recount_Tallies

This table contains a list of all candidate tallies that are computed during a recount. It exists because a local office may have some use for the original Tallies table in the event of suspected election fraud, and the original table would need to be intact and unmodified.

3.2.7.1. Schema

Recount_Tallies (*candidate_id*, *contest_name*, *election_date*, vote_count)

3.2.7.2. Description of attributes

Candidate_id: Primary key in integer format. Unique identification for a candidate or ballot option. Associated with the "candidate_id" attribute of Candidates (§3.1.3).

Contest_name: Primary key in string format. Unique name for a contest or ballot initiative. Associated with the "contest_name" attribute of Contests (§3.1.5).

Election_date: Primary key in date format. Unique date for an election. Associated with the "election_date" attribute of Elections (§3.1.1).

3.3. Functional Restrictions.

3.3.1. Overvotes disallowed.

For any *ballot_id* and *contest_name* in Votes, there can be no more than one *candidate_id*.

For any *ballot_id* and *contest_name* in Realtime_Votes, there can be no more than one *candidate_id*.

For any *ballot_id* and *contest_name* in Recount_Votes, there can be no more than one *candidate_id*.

3.3.1.1. Note

All InnoVote software products have built-in programming measures that do not allow overvotes to be generated. This is merely a fail-safe measure.

3.3.2. Negative tallies disallowed.

For any instance of Tallies, *vote_count* cannot be less than 0.

For any instance of Recount_Tallies, *vote_count* cannot be less than 0.

3.3.3. No votes for candidates that are not running.

For any instance of Votes, there can be no *candidate_id* and *contest_name* combination that does not exist in the Running table.

For any instance of Realtime_Votes, there can be no *candidate_id* and *contest_name* combination that does not exist in the Running table.

For any instance of Recount_Votes, there can be no *candidate_id* and *contest_name* combination that does not exist in the Running table.

3.3.3.1. Note

All InnoVote software products have built-in programming measures that do not allow voters to cast votes for persons that are not running for a particular office. In races where write-in candidates are allowed, the option “write-in” will be present in the list of candidate-contest combinations (the “Running” relationship set). This functional restriction is a fail-safe measure.

3.3.4. Write-in names allowed only when candidate choice is “write-in”

For any instance of Votes, if *candidate_id* is not “write-in”, the attribute “writein_name” must be NULL.

For any instance of Realtime_Votes, if *candidate_id* is not “write-in”, the attribute “writein_name” must be NULL.

For any instance of Recount_Votes, if *candidate_id* is not “write-in”, the attribute “writein_name” must be NULL.

3.3.4.1. Note

All InnoVote software products have built-in programming measures that do not allow the “writein_name” field to be used unless a Voter has chosen to vote for a write-in candidate. This is a fail-safe measure.

3.3.5. Deletion of an item permitted only if no other table entry references it

All of the tables must disallow “cascading;” i.e., deletion of a table entry may be allowed only if no other tables contain entries that reference it.

4. Encryption

4.1. Encrypted Tables.

The sensitive nature of certain data stored in InnoVote Databases requires that the tables containing such data be encrypted. Manipulation of any part of an InnoVote Database could have dire consequences for the integrity of elections; however, as is stated in §3.1, the database will be configured to disallow deletion of entity sets whose attributes are used in any relationship sets. This policy will prevent many types of tampering.

The user authentication procedure for the Database (§5) should prevent unauthorized access to sensitive data. However, in the event that the security mechanisms of the Database should become compromised and the Database be open to unauthorized SQL querying, the tables containing individual vote data would require extra protection from tampering. Tampering with votes can be done in three ways: adding new votes, modifying existing votes, or deleting votes. After such tampering occurred, the attacker would need to modify the Tallies table so that the built-in fraud-checking mechanisms of ReliaVote PE and ReliaVote CS (refs. [6] and [7]) would fail to detect the tampering.

Encrypting the entire table prevents all three forms of tampering, as opposed to encrypting individual table entries (which does *not* prevent deletion of votes). If an attacker were to gain direct access to the Database and execute SQL queries on it, the only fraudulent operation he could perform on a vote table would be to delete the entire Votes (or Realtime_Votes or Recount_Votes) table, a fraud that would *instantly* be detectable by ReliaVote software. It is for these reasons that the Votes, Realtime_Votes, and Recount_Votes tables will be encrypted.

4.2. Cryptosystem.

The cryptosystem can be either symmetric or asymmetric. The asymmetric encryption scheme provides extra security but requires more complicated key management.

4.2.1. Symmetric encryption

In a symmetric encryption system, each table is encrypted with a secret key. The keys are not chosen by a human user and should not be known except by the software. When an InnoVote software product needs to modify a vote data table, it obtains the key and decrypts the table. As is stated in the Functional Design documents for each InnoVote software product, every time a modification is made to one of these three tables, a new key is generated for the re-encryption.

4.2.2. Asymmetric encryption

In an asymmetric encryption scheme, each table has its own public/private key combination. Each *valid* operation of an InnoVote software product that needs to modify a table has a public/private key combination as well, and it is the only operation that can use the private key. When an operation needs to modify a vote table, it sends a signal to the database management system containing its own private key and the public key of the table that it needs to use. The database management system verifies that the operation is permitted to modify the table by checking for the existence of a public key that matches the private key. It then decrypts the table using the table's private key and grants the operation access to the table.

As with the symmetric encryption scheme, after the operation is finished modifying the table, the database management system generates a new public/private key pair for the table. It re-encrypts the table with the new private key.

4.3. Key Management.

4.3.1. Symmetric encryption

The key management for the symmetric cryptosystem (§4.2.1) requires that the key to each encrypted table be stored in a secure location. Only certain operations will be permitted to obtain keys. The database management system must be able to authenticate software operations to determine that they should be allowed to decrypt a table.

4.3.2. Asymmetric encryption

The key management for the asymmetric cryptosystem (§4.2.2) requires the storage of public keys for tables *and* validated software operations. It uses a Kerberos-type protocol, in which an operation requesting a public key must authenticate itself to the key management system. This authentication shall consist of the DBMS' detection that the operation originated with an InnoVote software product; i.e., the DBMS trusts the software products.

The asymmetric cryptosystem also requires that the private keys of the tables be stored in a secret location. The private keys of the operations can be stored either by the software product or by the database management system; unlike the tables' key pairs, the key pairs for software operations will not need to be changed unless they are compromised.

5. User Privileges

5.1. DBMS.

The database management system will have full privileges to the Database tables. It will be able to perform any operation, including deletions, changes, adds, and decryption. It must not be possible for any person or external software product to gain this privilege level.

5.2. InnoVote Software Products.

InnoVote software products that will use a Database include SecureDRE, CardReader Software, ReliaVote Precinct Edition, and ReliaVote Central Server. These products need to have access to a Database located on the systems on which they are running. These products should be allowed to make any change to entries within tables but should not be able to change the underlying relational schema of the Database or change their own or other users' privilege levels. There are two possible configurations for granting user privileges to these products.

5.2.1. Software as user

The first configuration involves granting "user" status to the particular installation of one of these software products that resides on a machine. The software will have to authenticate itself to the DBMS, which will then grant it full access to make any modification to entries of a table.

This configuration works best with a symmetric encryption scheme (§4.2.1) but could be used with an asymmetric encryption scheme (§4.2.2) as well.

5.2.2. Software operation as user

The second configuration is more complex but provides better security from compromising of the software products. This configuration involves creating separate "users" for every software operation (described in the Functional Design documents, references [2], [6], [7], and [8]) that needs to use a table. Each operation needs to authenticate itself to the DBMS, which then grants it the proper privileges. Some operations need to read tables but not write to them (such as retrieving election data); others need to be able to write without reading existing table entries (such as casting votes); others need to do both (such as conducting a recount). Under this system, each operation would have a list of privileges associated with every Database table.

This database configuration provides the best security with an asymmetric encryption scheme (§4.2.2) but could be used with a symmetric encryption scheme (§4.2.1) as well.

Reference [3] shows an access matrix for all operations of InnoVote software products, as defined in §3.1, “Functional Requirements,” of the Functional Design document for each product. The diagram effectively details the necessary security policy for the implementation of this user model. In the diagram, it is assumed that the operations will be performed at System level and will modify only the database stored on a particular machine. No InnoVote product is able to remotely access the database of another InnoVote product.