Reference Source Code Implementation

API Developer's Guide

April, 2004

Contents

Chapter 1

Getting Started

1.1 Compiling and Running VHTi Applications

1.1.1 Prerequisites

Current build system requirements:

1. Windows 2000 Professional with Service Pack four. Older versions, such as Windows NT 4, will probably work, but have not been tested.
2. Microsoft Visual C++ 6.0
3. Cygwin (`http://www.cygwin.com/setup.exe`). Install *all* the packages.

1.1.2 Third-Party Libraries

We use the following third-party libraries:

1. libxml2 2.6.3

2. iconv 1.9.1
3. openssl 0.9.7c
4. zlib 1.1.4

Current versions of the above three libraries can all be found at

`http://www.zlatkovic.com/projects/libxml/binaries.html`.

We may substitute newer versions of the above libraries as they become available; this list reflects our current development and testing environment.

The libraries are distributed as ".zip" files. If you use the Cygwin "unzip" program to unpack them, be sure to make the unpacked DLLs executable by doing `chmod +x *.dll`.

Be sure to tell Visual C++ about the libraries by clicking Tools, Options, Directories. Also put the directories in which you've placed DLLs onto your PATH (right-click "My Computer", choose "Properties", "Advanced", "Environment Variables", "New").

1.1.3 Building

To build the VHTi examples, open the VSS workspace `VHTI.dsw`, and build the "_tests_and_samples" project.

1.1.4 Running

The easiest way to run the examples is through Visual Studio: right-click the appropriate project, choose "Set as Active Project", then press `Ctrl+F5`. If you want to see what is going on inside, and you probably will, then use the debugger. Refer to Microsoft's documentation for advice on setting breakpoints and watching variables.

1.2 Changes to VHTi

As VHTi version 1.5 is the first release of the API, there are no changes made within this document that may affect any previous release versions. For future releases, any changes to the API or its subsequent documentation set will be included within this section.

1.3 Bug Reports

As VHTi version 1.5 is the first public-domain release of the API, there are currently no bugs or issues to report.

Known Issues

There are currently no known issues to report with VHTi version 1.5.

Reporting an Issue

VoteHere appreciates your efforts in helping us identify and resolve issues and inaccuracies with our products, specifications and documentation. If you feel you have identified an issue with the VHTi API or documentation set, please proceed with the following steps for submitting the issue to the VoteHere support team:

1. Record the version number of the API or document you are referencing, and if documentation-related note the page and /or section number.
2. Record and document the issue as clearly and in as much detail as possible.
3. Record your name, company name, and a telephone number where you can be reached during normal business hours.
4. Contact VoteHere using one of the following methods:

- **Email:** support@votehere.net
- **Fax:** 1.425.450.2861
- **Phone:** 1.888.457.6863

Customer Support Hours of Operation

Monday–Friday 9:00 a.m. to 5:00 p.m. USA Pacific Standard Time, excluding US national holidays.

1.4 Examples

The following examples are provided to show how to use VHTi. Except for the *Election* example, they are constructed as Windows Console Applications, and will show status messages in the output window. Usually a statement about what operation is being performed is followed by a success or failure message about that operation. When the sample program is done, it will report that it is FINISHED EXECUTING.

- *Election* exercises VHTi through a series of Bash shell and Perl scripts.
- *KeyShare* exercises the key sharing and election configuration functions.
- *Tabulation* exercises the tabulation functions.
- *VoterVerify* exercises the voting machine functions with voter-verification.
- *Voting* exercises the voting machine functions.
- *PKI* exercises the public key infrastructure functions.

1.4.1 Election Example

The election sample exercises VHTi through a series of Bash shell and Perl scripts. It demonstrates the VHTi protocol from election configuration through audit, including pre-election configuration, voting, post-election tabulation, and post-election audit. See the README for a full description.

1.4.2 Key Sharing Example

The key sharing sample demonstrates the process that each authority will follow in order to generate his SecretShare[1]. Election configuration is also demon-

strated in this sample by showing how to generate CryptoGroupParameters which is needed throughout VHTi.

Secret shares are used during the Tabulation process. The key sharing sample begins with the generation of KeyGenParameters and the actual Authority objects. Some SeedParameters are necessary to make the KeyGenParameters. The sample provides default values for these: the number of Authorities, n, is 4; the threshold number of Authorities, t, is 3; and a seed to generate random numbers is 123. To change these, using Visual Studio's Project settings, set the "Program Arguments" to your n, t, and seed.

The sample code then generates SecretCoefficients and BroadcastValue values for each Authority. Given these, the PairwiseSecrets can be calculated, and then finally the KeyShareCommitment and SecretShare for each authority. The Key Share Commitment is verified upon creation.

1.4.3 Voting Example

The voting sample code uses as input a BlankBallotand a ClearTextBallot. Since the ballot encryption function requires a SignedBlankBallot, the sample signs the BlankBallot using the signing function from the PKI library. Although the sample doesn't demonstrate it, the encryption function (like all functions that take signed objects) verifies that signature, again using the PKI library, before doing its real work.

1.4.4 Voter Verification Example

The voter verification sample is similar to the *Voting* sample with additional voter verification added. The sample includes procedures for

1. Pre-election procedures for generating BallotSequenceNumbers, VoteVerificationKeys, and SigningPrivateKeys;
2. Election procedures for generating dictionary VoteVerificationCodes, VoteReceipts on the voting machine; and
3. Post-election to support VoteVerificationTrustees generation of VoteVerificationCodes from SignedVotedBallots to allow voters to verify that their voted ballot, as verified in the pollsite, is contained in the ballot box.

1.4.5 Tabulation Example

The tabulation sample code demonstrates an individual CommittedAuthority partially decrypting each RawVotedBallot in a RawBallotBox and placing the result into an AuthorityPartialDecrypt structure. After enough authorities have created their AuthorityPartialDecrypts, the collection of AuthorityPartialDecrypts can be bundled into an AuthorityPartialDecrypts structure, which can then be combined with a single RawBallotBox to create a PartiallyDecryptedBallotBox. The combine decrypt function takes this structure as an argument, along with a single SignedBlankBallot, and simultaneously

1. Verifies that the collection of CommittedAuthoritys that is presented in the PartiallyDecryptedBallotBox are legitimate, in the sense that their KeyShareCommitments constitute a valid share of the ElectionPublicKey that is presented in the SignedBlankBallot. (Note: This check alone does not prevent a malicious entity from impersonating one or more of the CommittedAuthoritys, since CommittedAuthority data is public information. However, an impostor would not be able to produce data that also passes the next check, 2. Hence the combination of this verification step with the following one are sufficient to establish the authenticity of the source of partial decryption data.)

2. Verifies all AuthorityPartialDecrypt validity proofs (i.e. checks the integrity of each partial decryption in the AuthorityPartialDecrypts)

3. Assuming that all validity proofs are correct, completes decryption of all the individual answers contained in the RawBallotBox, by properly combining the partial decryption information

1.4.6 Public Key Infrastructure Example

The PKI sample code demonstrates basic elements of cryptography: creating a Signature on a document, verifying a Signature, encrypting a document, and decrypting a document. A GeneralPurposePublicKey and a GeneralPurposePrivateKey are needed for these operations, so the sample begins by generating them. A simple XML document containing the phrase "Hello World" is passed to the signing operation along with a GeneralPurposePrivateKey. The signing operation creates a hash of the document; this hash is what is actually signed. The output XML structure is a Signature. The signature verification function

takes the original document and Signature, re-computes the hash, and compares it to the hash in the Signature.

The encryption sample takes a ConstCharStar and a GeneralPurposePublicKey and places the encrypted data into an EncryptedData. Decryption takes the EncryptedData plus a GeneralPurposePrivateKey, and returns the original ConstCharStar.

Chapter 2

Fundamentals of VHTi

2.1 VHTi Election Fundamentals

Stripped to its essence, the most fundamental requirement of an election is that it produces a published count for the choices of eligible voters, which is then approved by an overwhelming majority of the population. Typically, this process is divided into two stages: Casting and Tabulation. To maintain the electorate's belief in and willingness to be bound by the outcome of an election, both voters and independent observers must verify these casting and tabulation stages. Voter Verification ensures that each vote is counted-as-cast. Open Results Verification ensures that all voted ballots are tabulated correctly. Taken together, Voter Verification and Open Results Verification provide total verification of an election—that is, Election Verification.

Casting is the recording of choices on some media format, encoded according to a publicly documented set of rules. This recorded data is generally referred to as the voter's *"voted ballot"* .

Tabulation is the counting of the collection of voted ballots (that is, "ballot box"). The ballot box could be a stack of pieces of paper kept in a locked vault under constant watch—in which case one might argue that the ballot box is not publicly approved, but only approved by a set of election trustees who *represent*

the public. The ballot box could also be posted on the Internet—in which case, one could argue, it is much closer to being publicly approved since access to the ballot box is granted much more broadly.

Voter Verification ensures that your voting intent is accurately recorded and contained in the ballot box for tabulation—that is, your ballot is counted as cast. Since only you know your intent, only you can accomplish Voter Verification. Specifically, it involves two distinct checks:

1. You must be able to satisfy that your voted ballot is properly recorded,
2. You must be able to satisfy that your voted ballot is contained in the public ballot box.

The goal of Open Results Verification is to produce an irrefutable tally from the public ballot box. Open Results Verification picks up where Voter Verification leaves off in that it provides an indisputably accurate sum (that is, election result) from the ballot box of votes, according to a publicly accepted counting process. Any scrutinizer (independent observer, election official, party observer, non-governmental organization, press, etc) can verify election results. Open Results Verification should involve three distinct checks, which must be accomplished without violating your secret ballot:

1. Any scrutinizer must be able to satisfy that ballots come from legitimate voters,
2. Any scrutinizer must be able to satisfy that no voted ballots have changed since they were cast,
3. Any scrutinizer must be able to reproduce the election results.

Because the ballot box is stored digitally and available redundantly across the Internet, the precision of the election results can be verified by any and all who are interested.

The simplest election is a "show of hands" election where voters raise their hands to indicate their vote. It is Voter Verifiable because all voters know whether their hand is up or not (cast "yes" or "no"). It is Open Results Verifiable because everyone can form their own count of hands raised, and thus no one needs to rely on the accuracy and integrity of some special vote counter, or counters. "Show

of hands" elections, however, do not offer a secret ballot, and are not suitable for large elections.

Small scale, hand-counted paper voting systems – those where everyone involved can actually watch the ballots and ballot box all the way through tabulation – are Voter Verifiable because (1) all voters can check that their vote is recorded on paper as intended (they mark the paper themselves) and (2) they can carefully watch the ballot box to ensure that their vote is included for counting. It is Open Results Verifiable because everyone can watch and participate to ensure that (1) only legitimate voters cast ballots, (2) ballots are not changed, and (3) the final count is completely accurate.

Machine counted paper voting systems are not quite Voter Verifiable because although voters can check that their vote is recorded on paper as intended, they cannot check to make sure that their ballot is in the ballot box when counting begins. It is not Open Results Verifiable because voters must trust the accuracy and integrity of the ballot handlers and counters. The United States Presidential Election in 2000 showed that machine counted paper ballots could be lost or inadvertently changed after casting; ultimately, it took a decision by the U.S. Supreme Court to decide the election.

Current electronic voting systems are neither Voter Verifiable nor Open Results Verifiable. Voters cannot even verify that something that might be called a "ballot" is ever created, much less that (1) their ballot accurately represents their choices, or (2) that their vote is "in" something that might legitimately be called "the ballot box." Furthermore, there is essentially no transparency in the counting process, so nearly no one can have any confidence that the results that are finally produced by some counting machine are an accurate total of the ballot information used as input.

For a complete overview of the fundamentals of VHTi and Verifiable e-voting, please refer to the VoteHere white paper titled "Verifiable e-Voting" included on the VHTi SDK CD.

2.2 VHTi Library Fundamentals

2.2.1 Overview

The VHTi library is designed to provide functionality without restricting policy, as long as the basic security requirements of the protocol are met. This means that at the lowest level, the VHTi API does not describe how data are stored by the participants, nor does it describe how the data are to be transferred between participants, nor does it restrict the user interface presented by devices using VHTi. Clients to the library will be responsible for maintaining privacy by encrypting data, using SSL/HTTPS, etc. once the data is given to them.

In order to make storage and transportation as simple as possible, all VHTi data structures are encoded as XML formatted text, conforming to elements described in vhti.dtd. In many cases, applications built upon VHTi can ignore the contents of these structures, and just pass them to the next API function. A program that uses VHTi will often group the results of various function calls together and pass them as a vector (array) to another function. Even clients with no XML parsing facility can do this, through simple string manipulations: remove the XML declaration from the elements, concatenate them together, and wrap the resulting string with a new XML declaration and document tag.

All of the VHTi functions validate their inputs against vhti.dtd, which is compiled into the library. However, you might want to use a different DTD. In that case, use the SetDTDLocation function to present a DTD file before calling any functions that take XML as input.

Naming Conventions

In the include (.h) files themselves, types are named with `HumptyCase`. This is true for types described in the API reference, the glossary, and the DTD file. Variables, which are instances of data types, are `named_with_underscores`. Names of functions (which are instances of transformation types) are prefixed with "`VHTI`" and `VHTI_named_with_underscores`. Enumerations and error strings are all `UPPERCASE_WITH_UNDERSCORES`.

For the sake of readability, names of functions are presented in an alternate form in the text of document itself. This form is simply a HumptyCase version of the form found in the include file *without* the prepended "VHTI." For example, the .h file function, `VHTI_encrypt_ballot_pollsite` is written as

EncryptBallotPollsite in the text of the document.

Parameter Conventions

All data passed to or from the functions are passed as parameters. In the current implementation, these parameters are not optional. The function return values are reserved for error conditions, and will be discussed shortly. Data returned from a function ("out-mode" parameters) are placed into character pointers. Those pointers are returned via "`char **`" addresses. (Note that NULL is *not* an allowed out-mode parameter address. The VHTi library does not check pointer validity on these arguments.) Convenient typedefs are provided by the header files to suggest the expected DTD type. VHTI_free must be called on each returned value once the client is finished with it.

Functions sometimes fail. Failure is indicated by a non-zero return value. Every failed function call will set an internal error state. These can be queried with GetLastError and GetLastErrorNumber. These error strings and numbers are established in the VHTI_error.XML file, which allows for basic internationalization or custom error messages. This error file is to be supplied by the library client, since it may need to support multiple languages or a particular formatting convention in the error messages.

Functions which are called upon to perform validity checks on proofs may find that a proof is not valid. In this case, they will still return a zero return value, because the function itself completed correctly. Instead, a failure will be reported through a "result" out-mode parameter.

Implicit Conventions

There are a few conventions that we follow which are currently not fully documented outside of the code itself. The ones which we have identified include hashing data structures, and signing XML.

Sometimes when we hash a data structure we feed the full XML into OpenSSL's digest functions, while other times we feed the big-endian representation of the numbers into the digest functions. There is no real security implication, but for another implementation to interoperate with us would require careful analysis of our code in order to create compatible hashes. In subsequent versions of the library, this operation will be standardized and compartmentalized.

Our current implementation of XML signing treats the XML as a sequence of bytes (rather than some representation of a more abstract data) when it signs it. The implication is that if someone were to create a detached signature with VHTI_sign, then modify the XML in any way–even just adding whitespace within an element tag–the signature would no longer check. The XML Security specification handles this with canonicalization of the input XML. For our purposes we decided that was unnecessarily complicated. In the future, this might change.

In this document, the mappings, or correspondences that are created by the VoteVerificationTrustee(s) are refered to as VoteVerificationDictionary(s). This is out of sync with the terminology in some of the VHTi white papers and promotional literature where the same construct is refered to as a "*codebook*." Please keep this terminology departure in mind in that which follows.

Along the same lines, Authority(s) and VoteVerificationTrustee(s) both provide similar stewardship roles to the election, and in fact, it will often be the case that one individual or public entity will fill both roles simultaneously. In early versions of the document, it was assumend that an Authority would always fill both roles. As a result, there may yet be some places in the document where these two roles have not been distinguished with full clarity.

2.2.2 Election Configuration

Before any voting can occur, some basic information must be agreed upon in accordance with jurisdiction law or policy. This information must then be digitally signed, and hence authorized, by the required set of officials, which again is determined by jurisdiction law or policy.

This information is encapsulated in the following data structures:

1. One or more TrusteeDictionaryCommitmentsSet data structure(s)
2. One or more SignedBlankBallot data structure(s)

In the case of each of these data structures, there needs to be exactly one for each "precinct" or "ballot style." (See Precinct.)

The TrusteeDictionaryCommitmentsSet structures are each essentially just collections of DictionaryCommitments. Each DictionaryCommitment can be thought of as secret information "sealed" in an envelope which is in full public view. The secret information is both generated and "sealed" by the VoteVerificationTrustee(s) acting independently. The fact that all this information is made public serves as the foundation of the voting device audit – i.e. Voter Verification.

A SignedBlankBallot is simply a BlankBallot along with a requisite set of valid digital signatures as mandated by local election policy.

A BlankBallot describes the ballot that will be presented to all of the voters for one election. For our purposes, all of the voters in an election receive the same ballot. For an election with precincts, each precinct will be treated as its own election. It may be convenient to use the same CryptoElectionParameters for multiple elections in this scenario.

Put simply, a blank ballot contains a set of CryptoElectionParameters, which are so important that they will be explained in subsequent paragraphs, some questions, and the BallotTextStructure. The BallotTextStructure is whatever is convenient for the voting machine to present the ballot to a voter. The exact makeup of BallotTextStructure is not critical to the proper functioning of the protocol.

Each question in a ballot generally represents one race or issue (although for "preferential style" ballots, a separate question must be created for each ranked answer). For each question, the voter will select one answer. Each answer represents one candidate, or "ABSTAIN," or, when allowed, "WRITE_IN." Having an explicit abstain selection is necessary, because the length of the voted ballots for a precinct must all be the same, regardless of how many questions the voter opted to vote on.

Each question is assigned a QuestionReference, which is just an alphanumeric string. These must be unique in each election. Each answer is assigned an AnswerReference, which likewise must be unique per election. Additionally, each answer is also assigned an AnswerMark. These must also be unique per

election, and for proper security should be a big randomly chosen number.

For each question there is a QuestionTextStructure, and for each answer there is an AnswerTextStructure. These provide data convenient for the voting machine to have available for the proper presentation of the ballot to the voter. It is similar in content and usage to blank ballot BallotTextStructure.

CryptoElectionParameters are the "public key" of the election. They can come from anywhere, as long as they meet the criteria described in the glossary. A naive solution is for someone to just sit down and generate these numbers, but that would cause the one person who generated the numbers to have complete control over the election.

A better solution, from a trust point of view, is to use the Key Sharing protocol, where multiple authorities cooperate to generate a set of CryptoElectionParameters, where they each own a portion of the information required to decrypt ballots. As long as one tabulation authority (person with a share of the key) acts honorably, the integrity of the election is preserved.

2.2.3 Poll Site Voting

In today's world of beautiful graphical displays on everything from cellular phones to gas pumps, it is easy to forget that presentation is merely the look-and-feel of the device, while what is really important is what it does. VHTi is designed to address *only* "what it does". After using the BallotTextStructure from the ballot, the QuestionTextStructures from the questions, and the AnswerTextStructures from the answers, the voting machine application should have acquired a list of AnswerReferences which correspond to the voter's intent. These answer references must obviously match AnswerReferences defined in the BlankBallot, and furthermore there must be exactly one AnswerReference for each question defined in the BlankBallot.

There are six key events in the course of the VHTi voting protocol (interactive process) that are not generic.

VE1. Load VoteVerificationKeys on voting device.

- This is part of device initialization and thus occurs only once. The specifics are implementation dependent, and security sensitive.

VE2. Generate and display (voter specific) VoteVerificationDictionary.

VE3. Audit VoteVerificationDictionarys.

VE4. Generate the voter's SignedVotedBallot – the voter's "official digital ballot."

VE5. Generate and display the voter's VoteReceiptData.

VE6. Sign the voter's VoteReceiptData creating the voter's VoteReceipt.

(Examples of generic voting events are voter authentication, voter choice indication or selection, ballot casting, etc.)

We will not discuss the specifics of 1. As mentioned, these are implementation dependent. Care should be taken to make sure the VoteVerificationKey information is not leaked during the process, and it should be deleted when the poll site is closed. (Ideally, the only writable, non-volitile memory in the voting device is its ballot box storage. In this case, the VoteVerificationKey information would be deleted by turning the machine off.)

Generate and Display VoteVerificationDictionary

At the very beginning of each voting session, *before* the voter has been given any oportunity to interact with the ballot display, the voting device needs to generate a (voter specific) VoteVerificationDictionary. This is done by way of the function `VHTI_generate_vote_verification_dictionary`.

The VoteVerificationDictionary is a table, or mapping, which links each potential ballot answer to a VoteVerificationCode. As a final confirmation step in the voting process, the voter can check to see that his/her intended choice has been recorded by verifying that correct VoteVerificationCode (as indicated by the VoteVerificationDictionary) is provided as part of his/her VoteReceipt.

Audit VoteVerificationDictionaries

In order to ensure that VoteVerificationDictionarys are displayed accurately to all voters, a random sample can be taken by one or more observers, and checked using the set of TrusteeDictionaryCommitments for the election.

A set of BallotSequenceNumbers which are not being used to vote must first be identified. Then, each trustee participating in the audit generates a set of BSNRevealedDictionarySecrets, where, for each BallotSequenceNumber, there is exactly one secret for each BallotQuestion. This set generated by a single trustee is called a TrusteeRevealedDictionarySecrets structure. This step can be performed by calling the function `VHTI_generate_dictionary_secrets`.

Each collection of TrusteeRevealedDictionarySecrets can be checked against the RevealedDictionarySecrets generated by that same trustee, using `VHTI_check_dictionary_secrets`. If all of the secrets for all of the trustees check, then they can be combined to produce VoteVerificationDictionarys that should be identical to the ones which would have been presented to the voters at the beginning of the voting process. In fact, the observer will most likely initiate the voting process to obtain VoteVerificationDictionarys, but will not cast the ballots. The function to combine the TrusteeRevealedDictionarySecrets is `VHTI_combine_dictionary_secrets`.

Generate SignedVotedBallot

After receiving final confirmation from the voter (i.e. the voter has explicitly chosen to "cast" his/her ballot), the voting device should encrypt the corresponding ClearTextBallot (which is the temporary electronic record of the voter's choices) with `VHTI_encrypt_ballot_pollsite`. If everything goes well the function will yield a SignedVotedBallot. (By signing the enclosed VotedBallot, the voting device is testifying to the fact that it was properly created by that device, and approved (cast) by the voter operating that device.) The SignedVotedBallot is then the voter's official digital ballot and can be added to the SignedVotedBallots which the device should keep as its local ballot box.

Generate and Display VoteReceiptData

In order for a voter to *irrefutibly* verify that his/her ballot is *accurately* represented in the final set of digital ballots that are *publicly counted*, the voter must leave the voting booth with a VoteReceipt. This is essentially just an ordered list of VoteVerificationCodes – one per question – which presents each of the voter's chosen answers according to the "translation" (i.e. correspondence between BallotAnswer and VoteVerificationCode) given by the voter's

VoteVerificationDictionary.

The VoteReceiptData is computed by calling the funcion `VHTI_generate_vote_receipt_data`. Once computed, the VoteReceiptData should be presented to the voter for final review in some form that is *unalterable* by the voting device. (We say that the voting device *commits* the VoteReceiptData by so doing.) A printed piece of paper is a good way to accomplish this, but not the only way. *The voter should be asked to do a final confirmation of his/her choices by reviewing the presented VoteReceiptData.* This is much like the act of reviewing a printed ballot before placing it in a ballot box.

The BallotSequenceNumber should be the first element of VoteReceiptData committed. In fact, it is important to commit this *before* the voter makes any ballot selections. There is a technical reason for this requirement, however, it would be a lengthy digression to elaborate on it here.

The voter should be given the option to change the answers to those questions where the VoteVerificationCode is not as intended. If any changes are made, *the entire* VoteReceiptData should be re-presented as if for the first time.

Create VoteReceipt

Once the voter is satisfied with the contents of the VoteReceiptData, the voter should indicate final acceptence (i.e. choose to cast the ballot). The voting device must carry out this instruction from the voter by *sigining* the VoteReceiptData, thereby turning it into a VoteReceipt. (The difference between these two objects may seem small, but it is critical. The VoteReceiptData is just a list of VoteVerificationCodes, and thus could be created by anyone, anywhere. The VoteReceipt can only be issued by the voting device in the context of the election at hand.)

Creating the VoteReceipt from the VoteReceiptData is a simple matter of calling the function `VHTI_sign_receipt`.

2.2.4 Tabulation

Tabulation occurs in five steps.

Dispose Ballots

Ballot disposition (sometimes known as authentication) is the process of going through the SignedBallotBox and determining, for each SignedVotedBallot therein, whether or not it should be included in the final tally. The most obvious reason that a SignedVotedBallot might need to be "removed" from the tabulation would be that its signature is invalid (i.e. its signature does not match the public key for any legitimate voting device). However, there may be other reasons set by local election policy.

Any change in the SignedBallotBox, however, must be publicly justified in accordance with the jurisdiction's election law.

Once the SignedVotedBallots have been approved for tallying, they are then stripped of the BallotSequenceNumbers, thereby disassociating the ballots with individual voters.

The end result of this process, which can be performed by calling the function `VHTI_authenticate`, is a RawBallotBox which is then used as the input to the Mix/Shuffle process that follows.

Compute and Prove VoteVerificationCodes

In order for each voter to "see" that his/her voted ballot has been included *unaltered* in the set of ballots to be counted, the VoteVerificationTrustees must provide mathematical proof that the encrypted value – which is publicly available – is in fact an encryption of the choice that the voter made in the poll booth.

This proof is generated using the three functions

- `VHTI_generate_pre_verification_results`
- `VHTI_gen_partial_verification_results`
- `VHTI_check_vcode_partial_decrypts_and_combine`

Anonymize

Mixing, or shuffling, is the process that anonymizes the ballots. It is typically performed by the tabulation authorities - the same people who hold the partial decryption keys - but there is no requirement for that to be the case. As a matter of policy, anyone might be allowed to shuffle.

The entire mix process is composed of a sequence of individual mixes, each carried out by an independent "mixing authority." The input RawBallotBox to each individual mix is the the same RawBallotBox that is the output of the previous one.

The work of an individual shuffle is carried out with the function `VHTI_shuffle`.

The primary input, as mentioned, is a RawBallotBox, but along with it are some random bits (for seeding purposes), a SignedBlankBallot (for the CryptoElectionParameters) and a GeneralPurposePublicKey for validation purposes. The output is a new RawBallotBox and a ShuffleValidityProof. The ShuffleValidityProof is what allows an independent observer to confirm that the mixer "behaved honestly" – that is, did not change the clear text (unencrypted) values of the ballots in any way. In short, the ShuffleValidityProof assures *everyone* that both the input and output RawBallotBoxes will decrypt to the same set of clear text ballots, though not necessarily in any common sequence.

This assurance is provided by way of the corresponding verification function `VHTI_check_shuffle`. Any independent observer can use this function (in fact their own implementation of it if so desired) in order to confirm that any one of the individual mixes was carried out correctly.

Decrypt Ballots

Decrypting is done in two phases:

DCRPT1. Partial Decryption

DCRPT2. Combine Partial Decryptions

Partial Decryption

Since *no one* holds a secret key capable of decrypting ballots (the "election private key"), a collection of Authority s must cooperate to create "pieces" of the decryption of each post-mixing ballot. We call these pieces, AnswerPartialDecrypt s. Each AnswerPartialDecrypt is actually more than the Authority 's decryption piece itself, it also contains proof information – information that can be used by an independent observer to check that the Authority supplied the *correct* decryption piece.

Each AnswerPartialDecrypt is computed by each Authority by way of the function `VHTI_partial_decrypt`.

Similarly, an idependent observer can check the validity of each AnswerPartialDecrypt by using the function `VHTI_check_partial_decrypt`. (As with mix verification, the independent observer is free to use their own implementation of this function.)

Combine Partial Decryptions

Once all the necessary AnswerPartialDecrypt s have been computed, combining them is a straightforward public computation – i.e. it requires no additional secret information. This operation has been combined with checking of each AnswerPartialDecrypt in the function `VHTI_check_partial_decrypts_and_combine`.

The output of this fuction is a set of ClearTextBallot s which can be counted by any convenient method.

Publish

All the data needed to verify the election count is produced by the previous steps. The only thing remaining is to make the data public, which is achieved by publishing it as widely as possible.

It is important to understand that this does not mean merely publish the final results. It means publishing all intermediate data, including validity proofs, that are necessary for an independent observer to verify the correctness of each tabulation step.

Note that "all intermediate data" does not include private inputs such as RandomState and GeneralPurposePrivateKey data since these are not required for verification purposes.. In a future version of this document we will provide a formal list of exactly what data structures must be published to allow for end-to- end verification.

Of course, publication is a standard process so no VHTi library functions are necessary to support it.

Chapter 3

VHTi API Reference

3.1 Security Considerations

The DTD elements in the following list should be kept secret.

ClearTextBallot

EncryptionPrivateKey

GeneralPurposePrivateKey

PairwiseSecrets

PairwiseSecret

RandomBlock

RandomIJState

RandomSeedKey

RandomSeedKeys

RandomState

SecretCoefficients

SecretShare

An obvious exception is made for all ClearTextBallot data that is produced as the *output* of Tabulation. This data is published and hence completely public.

3.2 Election Configuration

3.2.1 Key Sharing

Key Sharing is the process by which a group of Authority s create an ElectionPublicKey whose corresponding secret key is "unknown." It is unknown in the sense that finding it requires the cooperated effort of at least t of the Authority s, where t is a parameter that is freely settable by election policy. The entire process can be broken down into the following sequence of operations:

KS 1. Generate Secret Coefficients

KS 2. Generate Broadcast Values

KS 3. Generate Public Key

KS 4. Generate Pairwise Secrets

KS 5. Generate Key Share Commitments

KS 6. Verify Key Share Commitments

It is assumed that the following election parameters have been agreed upon by the participants in advance:

1. *Number of Tabulation Authorities*: integer n
2. *Threshold Number of Tabulation Authorities*: integer t
3. *Election Modulus*: prime integer p

4. *Election Subgroup Modulus*: prime integer q with $q|(p-1)$
5. *Election Generator*: $g \in \mathbf{Z}_p$, $|g| = q$
6. *A Hash Function*: H
7. *Participant Identifiers*: AuthorityEvaluationPoint$_i \in \mathbf{Z}_q^*$

$$\text{CreateKeygenParameters}\left(\text{SeedParameters}, \text{RandomState}\right) \longrightarrow \left(\text{KeyGenParameters}, \text{RandomState}\right)$$

This is a *helper* function which will generate a set of the required election parameters. It is primarily supplied for the purposes of testing and evaluation. In practice, these parameters would be agreed upon by a cooperative process, not generated by a single entity.

$$\text{CreateAuthority}\left(\begin{array}{l}\text{KeyGenParameters}, \text{RandomState}\\ \text{GeneralPurposePublicKey}\end{array}\right) \longrightarrow \left(\begin{array}{r}\text{GeneralPurposePrivateKey}, \text{Authority}\\ \text{RandomState}\end{array}\right)$$

This is a *helper* function which will generate an Authority structure usable with the rest of the key sharing protocol.

If the GeneralPurposePublicKey parameter is NULL or the empty string, the function will generate a key pair, incorporate the GeneralPurposePublicKey into the Authority, and return the GeneralPurposePrivateKey. Otherwise it assumes that the owner of the GeneralPurposePublicKey already has the corresponding GeneralPurposePrivateKey safely stored somewhere, leaves the GeneralPurposePrivateKey parameter alone, and simply copies the GeneralPurposePublicKey into the Authority.

$$\text{GenerateSecretCoefficients} \left(\begin{array}{l} \text{KeyGenParameters} \ , \ \text{Authority} \\ \text{RandomState} \end{array} \right) \longrightarrow \left(\ \text{SecretCoefficients} \ , \ \text{RandomState} \ \right)$$

$$\text{GenerateBroadcast} \left(\begin{array}{l} \text{KeyGenParameters} \ , \ \text{SecretCoefficients} \\ \text{RandomState} \end{array} \right) \longrightarrow \left(\ \text{BroadcastValue} \ , \ \text{RandomState} \ \right)$$

$$\text{GeneratePublicKey} \left(\ \text{KeyGenParameters} \ , \ \text{BroadcastValues} \ \right) \longrightarrow \left(\ \text{ElectionPublicKey} \ \right)$$

This operation will check the broadcast values before generating the ElectionPublicKey. It is necessary to have broadcast values from each authority before the ElectionPublicKey can be generated.

$$\text{GenerateSecret} \left(\begin{array}{l} \text{KeyGenParameters} \ , \ \text{Authority} \\ \text{SecretCoefficients} \end{array} \right) \longrightarrow \left(\ \text{PairwiseSecret} \ \right)$$

where the `AuthFingerprint` (i.e. ID) contained in the SecretCoefficients parameter (this is the Authority who is *generating* the PairwiseSecret) will be the same as the `FromID` entry in the PairwiseSecret, and the `AuthFingerprint` contained in the Authority parameter (this is the Authority who is *receiving* the PairwiseSecret) will be the same as the `ToID` entry in PairwiseSecret.

Each authority will call this function n times to generate n pairwise secrets.

$$\text{GenerateCommitment}\left(\begin{array}{l}\text{KeyGenParameters}\ ,\ \text{Authority}\\ \text{BroadcastValues}\ ,\ \text{PairwiseSecrets}\end{array}\right) \longrightarrow \left(\ \text{SecretShare}\ ,\ \text{KeyShareCommitment}\ \right)$$

where the `AuthFingerprint` (i.e. ID) of the Authority parameter is the same as is the `ToID` field in all of the n PairwiseSecrets in the PairwiseSecrets parameter. This operation uses the broadcast values to check the recipient's PairwiseSecrets before generating its SecretShare and KeyShareCommitment.

$$\text{CheckCommitment}\left(\begin{array}{l}\text{KeyGenParameters}\ ,\ \text{Authority}\\ \text{BroadcastValues}\ ,\ \text{KeyShareCommitment}\end{array}\right) \longrightarrow \left(\ \text{CheckResults}\ \right)$$

where Authority is the authority whose commitment is being checked. Specifically, the `AuthFingerprint` of the Authority parameter is the same as the `AuthFingerprint` of the KeyShareCommitment parameter.

In addition to the key sharing library, some additional VHTi specific functionality is required to initialize an election.

3.2.2 Ballot Configuration

$$\text{GenerateAnswerMark}\left(\begin{array}{l}\text{CryptoGroupParameters}\ ,\ \text{AlphabetEncoding}\\ \text{RandomState}\end{array}\right) \longrightarrow \left(\ \text{AnswerMark}\ ,\ \text{RandomState}\ \right)$$

Current version of the library supports only RandomState or RandomIJState = NULL.

$$\text{GenerateBlankBallot}\left(\begin{array}{l}\text{BallotSkeleton}\ ,\ \text{CryptoElectionParameters}\\ \text{AlphabetEncoding}\end{array}\right) \longrightarrow \left(\ \text{BlankBallot}\ \right)$$

3.2.3 Dictionary Generation

$$\text{GenerateVvk}\left(\ \text{RandomState}\ \right) \longrightarrow \left(\ \text{VoteVerificationKey}\ ,\ \text{RandomState}\ \right)$$

$$\text{GenerateBsns} \left(\begin{array}{l} \text{ElectionID} \ , \ \text{PrecinctID} \\ \text{int} \ , \ \text{int} \\ \text{RandomState} \end{array} \right) \longrightarrow \left(\begin{array}{r} \text{BallotSequenceNumbers} \ , \ \text{BallotSequenceNumbers} \\ \text{RandomState} \end{array} \right)$$

The first BallotSequenceNumbers returned is to be used by *"authorized"* voters. The second BallotSequenceNumbers returned is to be used by *"provisional"* voters. The first and second `int` arguments are the number of BallotSequenceNumbers to be included in each of these.

$$\text{GenerateVvdictCommits} \left(\begin{array}{l} \text{Authority} \ , \ \text{GeneralPurposePrivateKey} \\ \text{VoteVerificationKey} \ , \ \text{BlankBallot} \\ \text{BallotSequenceNumbers} \end{array} \right) \longrightarrow \left(\text{SignedTrusteeDictionaryCommitments} \right)$$

Note that the GeneralPurposePrivateKey parameter must be the *unique* private key corresponding to the Certificate component of the Authority parameter. Calling the function with a mismatch between Authority and GeneralPurposePrivateKey will generate an appropriate ErrorStructure.

The voter verification protocol requires that each Authority (VoteVerificationTrustee) "publish" all of its TrusteeDictionaryCommitments relating to the election in question prior to the start of vote casting. The exact requirements for publication are a matter of public election policy set by the jurisdiction.

3.3 Poll Site Voting

The model for poll site elections differs from the model for remote (network) elections in three ways that affect system design:

Authentication Poll site elections rely on "face-to-face" authentication of prospective voters by human poll workers. While this has the advantage of eliminating the need for secure digital authentication of voters, it places a rather large amount of power over election results in the hands of the poll workers, who are even more prone to mistakes than malicious actions.

Coercion Poll site election requirements have historically been more stringent with respect to the issues of *vote buying* and *vote coercion* than remote (e.g. vote-by-mail) election requirements.

Client/Server Poll site electronic equipment has, for logistical reasons (i.e. ease of set-up) generally required that the voting machine (or voting client) and voting server be implemented on the same device. As a result, information techniques that leverage the seperation of these two functions to independently managed devices are of no value.

As a result, the following voter verification strategy is employed:

Pre-Election

1. BallotSequenceNumbers are generated by the election configuration application. A BallotSequenceNumber roughly corresponds to a blank ballot identifier.

2. VoteVerificationKeys are generated by each VoteVerificationTrustee through the Dictionary Generation procedure. The VoteVerificationKeys are used to generate the VoteVerificationDictionarys on the voting machines at vote-time.

3. Using its VoteVerificationKey and a known BallotSequenceNumbers structure, each VoteVerificationTrustee generates a TrusteeDictionaryCommitments structure. In principle, this is done *once* for each Precinct. However, if there is a requirement for seperate "provisional" BallotSequenceNumbers, a second TrusteeDictionaryCommitments must be generated for each Precinct which has such a requirement. Thus, assuming there are t VoteVerificationTrustees, then before the start of the voting, there should be, for each Precinct, t *published* "authorized" TrusteeDictionaryCommitments and possibly also t *published* "provisional" TrusteeDictionaryCommitments.

4. SigningPrivateKeys are generated so each voting machine can digitally sign each ballot.

5. Each voting machine is initialized with BallotSequenceNumbers, VoteVerificationKeys, and a SigningPrivateKey.

Election

1. Assign an arbitrary "unused" BallotSequenceNumber.

2. Generate the VoteVerificationDictionary corresponding to this BallotSequenceNumber. The VoteVerificationDictionary should be committed before voting. What constitues satisfactory form of committment is a matter of jurisdiction policy, and is left to the application.

3. Create a SignedVotedBallot from voter input (choices).

4. Generate VoteReceiptData for voter-verification against the VoteVerificationDictionary.

5. Sign VoteReceiptData for the voter to take from the pollsite.

Post-Election

1. For each voted BallotSequenceNumber and each QuestionReference, the VoteVerificationTrustees mutually generate a VoteVerificationCode and cryptographically prove its correctness relative to the published TrusteeDictionaryCommitments. This process is akin to Tabulation and can be executed in conjunction with it. In addition to its SecretShare, each VoteVerificationTrustee must also have its VoteVerificationKey (which is kept secret through the process). The combination of VoteVerificationCode *and corresponding cryptographic proof* allow each voter to simultaneously verify

   (a) That the voter's ballot has been recorded (cast) as the voter intended at the time of voting. (*"Cast as Indended"*)

   (b) That the voter's ballot has the desired effect on the final election count. (*"Counted as Cast"*)

2. For a set of unvoted BallotSequenceNumbers and each QuestionReference, the VoteVerificationTrustees each generate secrets using the same procedure as GenerateVvdictCommits. These BSNRevealedDictionarySecrets can be checked against the VoteVerificationTrustee's TrusteeDictionaryCommitments. If all of the BSNRevealedDictionarySecrets from all of the VoteVerificationTrustees validate, they can be combined to produce VoteVerificationDictionarys. By comparing these VoteVerificationDictionarys against those generated by the pollsite voting machine for each of the unvoted BallotSequenceNumbers in the set, confidence in the accuracy of the dictionaries can be achieved.

3.3.1 Initializing Voting Machines

Each of the VoteVerificationKeys used by each VoteVerificationTrustee in the process of Dictionary Generation must be installed on the voting device prior to the start of vote casting. The procedure for doing this is device dependent. The procedure should also be carefully controlled to prevent leakage of any of the VoteVerificationKey information before, during, or after the election.

Functions in the following subsections depend on the availability to the voting device of a VoteVerificationKeys object which contains exactly one VoteVerificationKey belonging to each VoteVerificationTrustee.

In addition, each voting device must have at least one SigningPrivateKey installed on the voting device for the purpose of signing (authenticating) VotedBallots – i.e. for the purpose of creating SignedVotedBallots from VotedBallots. (Multiple SigningPrivateKeys may be used – for example to sign the VotedBallots from distinct Precincts – but are not required.) The procedure for doing this is application dependent.

In order to assure that all disputes are straightforwardly resolvable, it is important that all the SigningPublicKey(s) (or Certificates) cooresponding to the SigningPrivateKey(s) used for the purpose of signing VotedBallots (within the scope of the election being run) be "published" prior to the start of vote casting. The exact requirements for publication are a matter of public election policy set by the jurisdiction.

Great care must be taken to protect the secrecy of all GeneralPurposePrivateKeys before, during, and after the election.

- If *all* of the VoteVerificationKeys are simultaneously compromised, it would be possible to confirm a voter's choices from the election verification transcript.
- If *any* of the SigningPrivateKeys are compromised, it would be possible to cast doubt on election integrity. This would be done by generating a large number of SignedVotedBallots that were not actually cast legitimately on the voting device or devices.

3.3.2 Dictionary Display

$$\text{GenerateVerificationCode} \left(\begin{array}{l} \text{VoteVerificationKeys} \ , \ \text{BlankBallot} \\ \text{BallotSequenceNumber} \ , \ \text{AnswerReference} \\ \text{int} \ , \ \text{AlphabetEncoding} \end{array} \right) \longrightarrow \left(\text{VoteVerificationCode} \right)$$

The `int` parameter is used to determine how long (*in bits*) the confirmation strings will be. (A reasonable range for it is 16 - 32.)

Formatting and display of the dictionary with corresponding candidate information is left to the application.

3.3.3 Dictionary Audit

$$\text{GenerateDictionarySecrets} \left(\begin{array}{l} \text{SignedVotedBallots} \\ \text{Authority} \\ \text{GeneralPurposePrivateKey} \\ \text{VoteVerificationKey} \\ \text{BlankBallot} \\ \text{BallotSequenceNumbers} \end{array} \right) \longrightarrow \left(\text{TrusteeRevealedDictionarySecrets} \right)$$

The SignedVotedBallots are used to eliminate the voted BallotSequenceNumbers from the set, leaving unvoted BallotSequenceNumbers. These unvoted BallotSequenceNumbers can be used to generate VoteVerificationDictionarys at the pollsite, which are checked against the output of the function CombineDictionarySecrets.

$$\text{CheckDictionarySecrets} \left(\begin{array}{l} \text{TrusteeRevealedDictionarySecrets} \\ \text{SignedTrusteeDictionaryCommitments} \\ \text{GeneralPurposePublicKey} \\ \text{BlankBallot} \end{array} \right) \longrightarrow \left(\text{CheckResults} \right)$$

$$\text{CombineDictionarySecrets}\left(\begin{array}{l}\text{TrusteeRevealedDictionarySecrets}\\ \text{BlankBallot}\\ \text{int}\\ \text{AlphabetEncoding}\end{array}\right) \longrightarrow \left(\text{VoteVerificationDictionaries}\right)$$

3.3.4 Voted Ballot Encryption

$$\text{EncryptBallotPollsite}\left(\begin{array}{l}\text{ClearTextBallot}\ ,\ \text{BlankBallot}\\ \text{BallotSequenceNumber}\ ,\ \text{RandomState}\\ \text{GeneralPurposePrivateKey}\end{array}\right) \longrightarrow \left(\text{SignedVotedBallot}\ ,\ \text{RandomState}\right)$$

This function does some basic consistency checking on each ClearTextBallot that is passed in. First, it checks that the number of AnswerReferences in the ClearTextBallot corresponds to the number of BallotQuestions in the BlankBallot. It should be emphasized that VHTi treats every answer as belonging to a unique question; in other words, the total number of answers on a ballot will always equal the total number of allowed selections. An undervote, or abstaining from a selection, is acknowleged by VHTi as a unique answer, and the function caller is responsible for pointing out the missed selection to the voter, if desired. It is also the function caller's responsibility to 'translate' any ballot questions which allow the voter to 'Choose n' selections, where $n > 1$, into n separate questions. Further, VHTi does *not* prohibit the voter from choosing the same selection for some or all n positions; however, this is easily discovered after tabulation and election policy must dictate procedures for handling this condition.

Second, this function will report an error, and halt ballot encryption, if there is no AnswerReference in the BlankBallot which corresponds to an AnswerReference in the ClearTextBallot. Thus, only valid AnswerMarks will be used to produce the ElGamalPairs in the resulting SignedVotedBallot.

3.3.5 Receipt Display and Storage

GenerateVoteReceiptData (SignedVotedBallot , VoteVerificationKeys
BlankBallot , BallotSequenceNumber
ClearTextBallot , int
AlphabetEncoding) ⟶ (VoteReceiptData)

As with Dictionary Display, the `int` parameter is used to determine how long (*in bits*) the confirmation strings will be. (A reasonable range for it is 16 - 32.)

SignReceipt (VoteReceiptData , GeneralPurposePrivateKey) ⟶ ( )

3.4 Tabulation

The tabulation process, $Tabulate$, is *conceptually* defined by the function signature

$$\text{Tabulate}\left(\begin{array}{l}\text{SignedVotedBallots}\ ,\ \text{VoterRoll}\\ \text{BlankBallot}\end{array}\right) \longrightarrow \left(\ \text{ElectionResults}\ \right)$$

Since, in general, tabulation is effected by multiple, independent entities acting in cooperation, no VHTi function implements it directly. As discussed in the previous chapter, it is composed of five stages.

The functions used to carry out these stages are summarized as follows.

3.4.1 Dispose Ballots

$$\text{Authenticate}\left(\begin{array}{l}\text{SignedVotedBallots}\ ,\ \text{VoterRoll}\\ \text{BlankBallot}\end{array}\right) \longrightarrow \left(\ \text{RawBallotBox}\ \right)$$

Authentication should be careful to check EID consistency between SignedVotedBallot and SignedBlankBallot.

3.4.2 Compute and Prove VoteVerificationCodes

$$\text{GeneratePreVerificationResults} \left(\begin{array}{l} \text{VoteVerificationKey} \\ \text{SignedVotedBallots} \\ \text{SignedBlankBallot} \\ \text{GeneralPurposePublicKey} \end{array} \right) \longrightarrow \left(\text{PreVerificationCodeBox} \right)$$

Each Trustee produces a PreVerificationCodeBox by transforming each AnswerMark into an unhashed VoteVerificationCode inside of the ElGamalPair. The PreVerificationCodeBox contains a RawBallotBox structure, and the RawVotedBallots inside have had their original contents replaced by these transformed values.

$$\text{GeneratePartialVerificationResults} \left(\begin{array}{l} \text{PreVerificationCodeBoxes} \\ \text{SignedBlankBallot} \\ \text{GeneralPurposePublicKey} \\ \text{CommittedAuthority} \\ \text{SecretShare} \\ \text{RandomState} \end{array} \right) \longrightarrow \left(\text{AuthorityPartialDecrypt} \ , \ \text{RandomState} \right)$$

Each Trustee takes the collection of PreVerificationCodeBoxes from *all* of the Trustees, and performs his own partial decrypt. The function internally generates a combined RawBallotBox from all of the PreVerificationCodeBoxes, which is then passed to the PartialDecrypt function, to produce an AuthorityPartialDecrypt of VoteVerificationCodes.

$$\text{CheckVcodePartialDecryptsAndCombine} \left(\begin{array}{l} \text{PreVerificationCodeBoxes} \\ \text{AuthorityPartialDecrypts} \\ \text{SignedVotedBallots} \\ \text{SignedBlankBallot} \\ \text{GeneralPurposePublicKey} \\ \text{int} \\ \text{AlphabetEncoding} \end{array} \right) \longrightarrow \left(\text{VoteVerificationStatements} \right)$$

Now an Authority can use the collection of PreVerificationCodeBoxs to call the CheckPartialDecryptsAndCombine function, and use the output to generate a VoteVerificationStatement for each BallotSequenceNumber in the SignedVotedBallots.

3.4.3 Anonymize

$$\text{Shuffle} \left(\begin{array}{ll} \text{RawBallotBox} , & \text{RandomState} \\ \text{SignedBlankBallot} , & \text{GeneralPurposePublicKey} \end{array} \right) \longrightarrow \left(\begin{array}{r} \text{RawBallotBox} , \text{RandomState} \\ \text{ShuffleValidityProof} \end{array} \right)$$

The GeneralPurposePublicKey argument is used for checking the authenticity of the SignedBlankBallot argument, which, as a consequence, checks the authenticity of the CryptoElectionParameters used in the computation.

In a subsequent version of the library, the GeneralPurposePublicKey argument will be changed to a Certificate argument instead. This will give better integration with the PKI which underlies the election protocol.

The output of this computation should be *signed* so that accountability can be indisputably assigned in the case that later verification of the proof fails. The next version of the library will support this.

$$\text{CheckShuffle} \left(\begin{array}{l} \text{RawBallotBox} \ , \ \text{RawBallotBox} \\ \text{SignedBlankBallot} \ , \ \text{GeneralPurposePublicKey} \\ \text{ShuffleValidityProof} \end{array} \right) \longrightarrow \left(\text{CheckResults} \right)$$

As with the preceding function, Shuffle, the GeneralPurposePublicKey argument is used for checking the authenticity of the SignedBlankBallot argument.

In a subsequent version of the library, the GeneralPurposePublicKey argument will be changed to a Certificate argument as in the case of the preceding function, Shuffle.

3.4.4 Decrypt

$$\text{PartialDecrypt}\left(\begin{array}{l}\text{RawBallotBox}\ ,\ \text{SignedBlankBallot}\\ \text{GeneralPurposePublicKey}\ ,\ \text{CommittedAuthority}\\ \text{SecretShare}\ ,\ \text{RandomState}\end{array}\right) \longrightarrow \left(\ \text{AuthorityPartialDecrypt}\ ,\ \text{RandomState}\ \right)$$

$$\text{CheckPartialDecrypt}\left(\begin{array}{l}\text{RawBallotBox}\ ,\ \text{AuthorityPartialDecrypt}\\ \text{SignedBlankBallot}\ ,\ \text{GeneralPurposePublicKey}\end{array}\right) \longrightarrow \left(\ \text{CheckResults}\ \right)$$

$$\text{CheckPartialDecryptsAndCombine}\left(\begin{array}{l}\text{SignedBlankBallot}\\ \text{GeneralPurposePublicKey}\\ \text{PartiallyDecryptedBallotBox}\end{array}\right) \longrightarrow \left(\ \text{ClearTextBallots}\ \right)$$

Again, in each of the preceding three functions, the GeneralPurposePublicKey argument is used for checking the authenticity of the SignedBlankBallot argument, and this will be changed to a Certificate argument in a subsequent version of the library.

This function needs to indicate an error condition when a decrypted answer is invalid – i.e. produces an AnswerMark that is not valid according to the SignedBlankBallot. This is currently done, but awkwardly. The result of such an error condition is that the corresponding ClearTextBallot that is produced will, itself, contain error text. This will then be noticed when the routine to tally AnswerMarks is run on the ClearTextBallots output argument. A better solution would be to provide a second output parameter that indicates the invalid decryptions that were found, including their position in the PartiallyDecryptedBallotBox. In addition, an `int` return value should indicate success status, which can be further described via the GetLastError mechanism. This approach will be used in subsequent versions of the library.

3.5 Public Key Infrastructure

Comprises a set of operations

1. Generate Keys
2. Encryption
   (a) Encrypt data
   (b) Decrypt data
3. Signatures
   (a) Sign data
   (b) Verify signature

The internal implementation of these operations is unimportant. Since it may be provided by other parties, we will only concern ourselves with the interfaces presented.

GenerateKeys (IdentInfo)

⟶ (GeneralPurposePrivateKey , GeneralPurposePublicKey)

Create a pair of keypairs which have some internal properties that allow implementing the Encryption and Signatures functionality. Note that the private keys are not encrypted, so they need to be stored with care.

Encrypt (GeneralPurposePublicKey , ConstCharStar)

⟶ (EncryptedData)

Encrypt given ConstCharStar with a GeneralPurposePublicKey. EncryptedData is not readable. EncryptedData likely contains more than simply the ciphertext: the identity of the holder of the private key, encryption method used, and date are all candidates for extra information.

Decrypt (GeneralPurposePrivateKey , EncryptedData)

⟶ (ConstCharStar)

Decrypt given EncryptedData with a GeneralPurposePrivateKey. If successful, resulting ConstCharStar will be equal to the ConstCharStar presented to the Encrypt operation.

Sign (GeneralPurposePrivateKey , ConstCharStar)

⟶ (Signature)

Create a detached signature on ConstCharStar, which could only have been created with knowledge of the appropriate SigningPrivateKey. Signature will at minimum contain some numeric proof, but will probably also contain information about the signor's identity, algorithm used, and date.

VerifySignature (GeneralPurposePublicKey , ConstCharStar
Signature)

⟶ (CheckResults)

Verify a signature on ConstCharStar. A true result indicates *only* that the function was able to complete; it does *not* necessarily mean that the signature was valid. If the result is true, you must also check that the CheckResults contains the string "Success"; that then indicates that the Signature was on ConstCharStar, and was created with the SigningPrivateKey that corresponds to SigningPublicKey.

3.6 Support Functions

The VHTi Library also supplies several non-cryptographic support functions.

3.6.1 XML Parsing and Manipulation


```
SetDTDLocation ( const char * path )
                                        ⟶  ( int )
```

```
Validate ( const char * datatype , const char * data )
                                        ⟶  ( int )
```

3.6.2 Memory Management

```
Dup ( const char * thing )
                                        ⟶  ( char * )
```

```
Free ( const char * thing )
                                        ⟶  ( int )
```

```
FreeAll ( void )
                                        ⟶  ( int )
```


This is a handy function for lazy programmers, or when there is plenty of available memory. It clears and then frees any memory which was allocated by VHTi and then returned to the caller. This should only be freed once the library caller is sure that it is no longer referencing any old returned values.

```
FreeWithoutClearing ( const char * thing )
                                        ⟶  ( int )
```

This is a function to free memory allocated by the library without first zeroing out the contents. We encourage all users to call the other memory freeing functions instead, since the only drawback is a very small performance cost.

3.6.3 Random and Pseudo-Random Number Generation

```
GenerateRandomState ( RandomSeedKey )
                                    ⟶  ( RandomState )
```

```
GetBits ( RandomState , int )
                                ⟶  ( RandomState , RandomBits )
```


GetPermutation (RandomState , int)

⟶ (Permutation , RandomState)

3.6.4 Error Handling

GetLastError (const char ** error_text)

⟶ (int)

The last error value is stored in a global variable, like the C library. It is good practice to retrieve this value as soon as the error is reported. Retrieving the last error also clears the last error.

GetLastErrorNumber (void)

⟶ (int)

This function will only return an interesting number if one is assigned in the error XML files. By default, it normally just returns -1.

Chapter 4

Abbreviated Alphabetic Function Listing

Function Name	Header File
Alloc	support.h
Authenticate	auth.h
CheckCommitment	check_comm.h
CheckDictionarySecrets	check_dictionary_secrets.h
CheckPartialDecrypt	check_partial_decrypt.h
CheckPartialDecryptsAndCombine	check_pds_and_combine.h
CheckShuffle	check_shuffle.h
CheckVcodePartialDecryptsAndCombine	check_vc_pds_and_combine.h
CheckXml	sign.h
CombineDictionarySecrets	combine_dictionary_secrets.h
CreateAuthority	keyshare_util.h
CreateKeygenParameters	keyshare_util.h
Decrypt	crypt.h
Dup	support.h
Encrypt	crypt.h

Function Name	Header File
EncryptBallotPollsite	enc_ballot_pollsite.h
Free	support.h
FreeAll	support.h
FreeWithoutClearing	support.h
GenerateAnswerMark	gen_answer_mark.h
GenerateBlankBallot	gen_blank_ballot.h
GenerateBroadcast	gen_broadcast.h
GenerateBsns	gen_bsns.h
GenerateCommitment	gen_comm.h
GenerateDictionarySecrets	gen_dictionary_secrets.h
GenerateKeys	genkeys.h
GeneratePartialVerificationResults	gen_pv_results.h
GeneratePreVerificationResults	gen_pre_verification_results.h
GeneratePublicKey	gen_pubkey.h
GenerateRandomState	random.h
GenerateSecret	gen_secrets.h
GenerateSecretCoefficients	gen_seccoeff.h
GenerateVerificationCode	gen_verification_code.h
GenerateVoteReceiptData	gen_vote_receipt_data.h
GenerateVoteVerificationDictionary	gen_vvdict.h
GenerateVvdictCommits	gen_vvdict_comm.h
GenerateVvk	gen_vvkey.h
GetBits	random.h
GetLastError	error.h
GetLastErrorNumber	error.h
GetPermutation	permutation.h
PartialDecrypt	partial_decrypt.h
PasswordDecrypt	crypt.h
PasswordEncrypt	crypt.h
SetDTDLocation	support.h
Shuffle	shuffle.h
Sign	sign.h
SignReceipt	sign_receipt.h
SignXml	sign.h
Validate	support.h
VerifySignature	sign.h

Appendix A

Glossary of VHTi Data Structures

These are "pseudo-asn1" structures that are used in the inputs and outputs of the API components.

Alphabet Encoding (AlphabetEncoding) : An xml object containing character data specifying the encoding format for the returned VoteVerificationCode. Initially, supported alphabets are DEC (Decimal or base 10), HEX (Hexadecimal or base 16), and BASE64 (base 64). Others may be available in the future.

Answer Mark (AnswerMark) : For the purpose of the cryptographic protocols, each answer (in the AnswerReference sense) must be randomly (or pseudo-randomly) assigned an element of the election's ElectionEncodingSubgroup.

$$\text{AnswerMark} \doteq \gamma_A \in \text{ElectionEncodingSubgroup}$$

The set of AnswerMarks corresponding to a single BallotQuestion must be must be *distinct*. That is, if A_1 and A_2 are both answers to the same BallotQuestion, then $\gamma_{A_1} \neq \gamma_{A_2}$. In order to effectively implement these properties, AnswerMarks are generated publicly as a SHA-1 of Election, Precinct, and AnswerReference data. (See Ballot Configuration.)

Answer Partial Decrypt (AnswerPartialDecrypt) : A data structure of cryptographic quantities representing the piece of information contributed by a *single* Authority for a *single* ElGamalPair contained in a *single* RawVotedBallot.

$$\text{AnswerPartialDecrypt} \doteq \left\{ \begin{array}{l} \text{ModularInt } Z \\ \text{ModularInt } c \\ \text{ModularInt } d \end{array} \right\}$$

This represents a valid partial decryption of a ElGamalPair, (X, Y), with respect to an Authority, A, with KeyShareCommitment, C, if and only if

$$c = H(g\,,\, C\,,\, X\,,\, Z\,,\, g^d C^c\,,\, X^d Z^c) \tag{A.1}$$

where H is the system secure hash function (SHA-1).

Answer Reference (AnswerReference) : A small integer, which, in the context of a fixed election is uniquely associated with a specific (question , answer) pair. Note for the sake of the cryptographic protocols, each ballot question must have its own spcial "ABSTAIN" answer reference.

Answer Skeleton (AnswerSkeleton) :

$$\text{AnswerSkeleton} \doteq \text{AnswerTextStructure}_1\,,\, \ldots\,,\, \text{AnswerTextStructure}_n$$

Answer Text Structure (AnswerTextStructure) : A specification according to a yet to be determined data standard of the human readable data (display text, title, shorthand name, etc.) associated with a ballot answer (perhaps complicated to support multiple languages, etc.).

Arbitrary C string (ConstCharStar) : Any 0-terminated C string.

Authority (Authority) : The data structure identifying an *Authority* who has, in advance of the election, been appointed to "oversee" the election is

$$\text{Authority} \doteq \left\{ \begin{array}{c} \text{Certificate} \\ \text{AuthorityEvaluationPoint} \end{array} \right\}$$

This structure also has an attribute, "AuthFingerprint", which is usually a small decimal number, which serves as a simple identification number.

Authority Evaluation Point (AuthorityEvaluationPoint) : A modular integer β where

$$\beta \in \mathbf{Z}_q^* = \mathbf{Z}_q - \{0\}$$

Authority Partial Decrypt (AuthorityPartialDecrypt) : The data structure representing the decryption information contributed by a *single* CommittedAuthority.

$$\text{AuthorityPartialDecrypt} \doteq \left\{ \begin{array}{c} \text{CommittedAuthority} \\ \text{BallotBoxPartialDecrypt} \end{array} \right\}$$

Authority Partial Decrypts (AuthorityPartialDecrypts) : An XML structure representing a *set* of one or more AuthorityPartialDecrypt. The order of the elements of this set is irrelevant.

$$\text{AuthorityPartialDecrypts} \doteq \{\text{AuthorityPartialDecrypt}_1, \ldots, \text{AuthorityPartialDecrypt}_t\}$$

Authority Set (AuthoritySet) : A set of Authorities.

$$\text{AuthoritySet} \doteq \{\text{Authority}_1, \ldots, \text{Authority}_t\}$$

Ballot Answer (BallotAnswer) : This data structure (XML) is needed for both for the blank ballot and for results reporting after tabulation. However, it is only connected to VHTi through the interface. We leave specification vague for now, and perhaps even leave the specification to a third party or standards organization. Roughly, it must have an AnswerMark which is *unique for the ballot in question*, an AnswerReference which is also *unique for the ballot in question*, and an AnswerTextStructure.

$$\text{BallotAnswer} \doteq \left\{ \begin{array}{c} \text{AnswerReference} \\ \text{AnswerMark} \\ \text{AnswerTextStructure} \end{array} \right\}$$

Ballot Box Partial Decrypt (BallotBoxPartialDecrypt) : A vector of BallotPartialDecrypts representing the information contributed by a *single* Authority for *all* the (properly sequenced) RawVotedBallots contained in a RawBallotBox.

$$\text{BallotBoxPartialDecrypt} \doteq (\text{BallotPartialDecrypt}_1, \ldots, \text{BallotPartialDecrypt}_B)$$

Ballot Partial Decrypt (BallotPartialDecrypt) : A vector of AnswerPartialDecrypts representing the information contributed by a *single* Authority for *all* the (properly sequenced) ElGamalPairs contained in a *single* RawVotedBallot.

$$\text{BallotPartialDecrypt} \doteq (\text{AnswerPartialDecrypt}_1, \ldots, \text{AnswerPartialDecrypt}_Q)$$

Ballot Question (BallotQuestion) : This data structure (XML) is needed for both the blank ballot and for results reporting after tabulation. The specification is left to the application. Roughly, it must have a question text structure (perhaps complicated to support multiple languages, etc.), and a vector of BallotAnswers. *It must* have one distinguished answer, "ABSTAIN". If it is to allow a write-in response, *it must also* have a second distinguished answer, "WRITE-IN".

$$\text{BallotQuestion} \doteq \left\{ \begin{array}{c} \text{QuestionReference} \\ (\text{BallotAnswer}_1, \ldots, \text{BallotAnswer}_Q) \\ \text{QuestionTextStructure} \end{array} \right\}$$

Ballot Questions (BallotQuestions) : A vector of L BallotQuestions

$$\text{BallotQuestions} \doteq (\text{BallotQuestion}_1, \ldots, \text{BallotQuestion}_L)$$

Ballot Sequence Number (BallotSequenceNumber) : A ballot identifier. The set of BallotSequenceNumbers for a given Election (or Precinct) can be considered equivalent to the set of "potential voters" (including *"provisional voters"*) in the Election (or Precinct).

Ballot Sequence Numbers (BallotSequenceNumbers) : A vector of V BallotSequenceNumbers

$$\text{BallotSequenceNumbers} \doteq (\text{BallotSequenceNumber}_1, \ldots, \text{BallotSequenceNumber}_V)$$

BSN Dictionary Commitments (BSNDictionaryCommitments) : A structure which represents all DictionaryCommitments for a *fixed* ElectionID, *fixed* PrecinctID, *fixed* VoteVerificationTrustee and *fixed* BallotSequenceNumber. That is, a collection of $l \geq 1$ DictionaryCommitments, where l is the number of QuestionReferences in the BlankBallot.

$$\text{BSNDictionaryCommitments} \doteq \left\{ \begin{array}{c} \text{BallotSequenceNumber} \\ (\text{DictionaryCommitment}_1, \ldots, \text{DictionaryCommitment}_l) \end{array} \right\}$$

BSN Revealed Dictionary Secrets (BSNRevealedDictionarySecrets) : A structure which represents a collection of $l \geq 1$ RevealedDictionarySecretvalues for each unvoted BallotSequenceNumber, where l is the number of QuestionReferences in the BlankBallot.

$$\begin{array}{c} \text{BSNRevealedDictionarySecrets} \doteq \\ \text{BallotSequenceNumber} \\ (\text{RevealedDictionarySecret}_1 , \ldots , \text{RevealedDictionarySecret}_l) \end{array}$$

Ballot Skeleton (BallotSkeleton) :

$$\text{BallotSkeleton} \doteq \left\{ \begin{array}{c} \text{ElectionID} \\ \text{PrecinctID} \\ \text{QuestionSkeleton} \\ \text{BallotTextStructure} \end{array} \right\}$$

Ballot Text Structure (BallotTextStructure) : Miscellaneous information – election name, titles, page and display info, etc., associated with the human readable elements of a ballot.

Blank Ballot (BlankBallot) : This structure needs to link together all the cryptographic and conventional information associated with the Election,

Precinct, and set of races, candidates and issues that are to be contested.

$$\text{BlankBallot} \doteq \left\{ \begin{array}{c} \text{ElectionID} \\ \text{PrecinctID} \\ \text{CryptoElectionParameters} \\ \text{BallotQuestions} \\ \text{BallotTextStructure} \end{array} \right\}$$

Broadcast Value (**BroadcastValue**) : A (random) element of ElectionEncodingSubgroup. These values are generated as part of Key Sharing, and are an essential component of the Pedersen dealerless secret sharing scheme.

Broadcast Values (**BroadcastValues**) : An XML string containing the BroadcastValue from each authority.

Certificate (**Certificate**) : A PKI, typically X.509, certificate.

Check Results (**CheckResults**) : An XML structure containing the results of checking or verification.

Cipher Text (**CipherText**) : A stream of encrypted bytes. In order to be decrypted, you also need a GeneralPurposePrivateKey, an InitializationVector, and an EncryptedSessionKey.

Clear Text Ballot (**ClearTextBallot**) : A direct representation, in the context of a fixed BlankBallot, of a voted ballot – i.e. set of voter choices. Constructed as a vector of $\nu \geq 0$ AnswerReferences.

$$\text{ClearTextBallot} \doteq (\text{AnswerReference}_1, \ldots, \text{AnswerReference}_\nu)$$

Clear Text Ballots (**ClearTextBallots**) : An XML structure containing one or more ClearTextBallot.

Committed Authority (**CommittedAuthority**) :

$$\text{CommittedAuthority} \doteq \left\{ \begin{array}{c} \text{Authority} \\ \text{KeyShareCommitment} \end{array} \right\}$$

Committed Authority Set (**CommittedAuthoritySet**) : A set of CommittedAuthority s

$$\{\text{CommittedAuthority}_1, \text{CommittedAuthority}_2, \ldots, \text{CommittedAuthority}_t\}$$

Committed Trustee : Currently a synonym for CommittedAuthority.

Committed Trustee Set : Currently a synonym for CommittedAuthoritySet.

Crypto Election Parameters (**CryptoElectionParameters**) : All configuration parameters required for execution of the election cryptographic operations:

$$\text{CryptoElectionParameters} \doteq \left\{ \begin{array}{c} \text{CryptoGroupParameters} \\ \text{CryptoTabulationParameters} \end{array} \right\}$$

Crypto Group Parameters (**CryptoGroupParameters**) : The set of necessary mathematical parameters that can be generated very early – prior to authority selection and Key Sharing.

$$\text{CryptoGroupParameters} \doteq$$

$$\left\{ \begin{array}{c} \text{ElectionModulus}\ (p) \\ \text{ElectionSubgroupModulus}\ (q) \\ \text{ElectionSubgroupMember}\ (\mathrm{g}) \end{array} \right\}$$

Crypto Tabulation Parameters (**CryptoTabulationParameters**) : The set of necessary mathematical parameters that are required before voting can begin (even before a BlankBallot can be completed), but are not known until during or after Key Sharing.

$$\text{CryptoTabulationParameters} \doteq$$

$$\left\{ \begin{array}{c} \text{ElectionPublicKey}\ (h) \\ \text{CommittedAuthoritySet}\ (\textit{All}\ n\ \textit{Tabulation Authorities}) \\ \text{TabulationThreshold}\ (t) \end{array} \right\}$$

Dictionary Commitment (DictionaryCommitment) :

An element of the ElectionEncodingSubgroup. Each DictionaryCommitment is used to irrefutably link VoteVerificationCodes to encrypted choices. For proper overall election usage, these must be constructed and *published* prior to voting. For a fixed Precinct, the full set of DictionaryCommitments is (multi-dimensionally) indexed by all possible triples: VoteVerificationTrustee, BallotSequenceNumber, QuestionReference.

$$\text{DictionaryCommitment} \doteq C \in \text{ElectionEncodingSubgroup}$$

Dictionary Question (DictionaryQuestion) : A component of a VoteVerificationDictionary which identifies the BallotQuestion by QuestionReference and contains a DictionaryVerificationCode corresponding to the indicated AnswerReference. If $A_1 \neq A_2$ are two distinct AnswerReferences which are both possible responses to the *same* QuestionReference, Q, then the VoteVerificationCode for A_1 *must be different* then the VoteVerificationCode for A_2. However, A_1 and A_2 may sometimes share the same VoteVerificationCode if they are possible responses to different QuestionReferences.

Dictionary Verification Code (DictionaryVerificationCode) : An element of a DictionaryQuestion which associates a particular VoteVerificationCode with the appropriate AnswerReference.

Election : Refers to all voting and tabulation issues within an umbrella jurisdiction, or political unit. Each election has one and only one CryptoElectionParameters structure associated with it. However, an Election can be subdivided into Precincts, which each have their own BlankBallot, possibly, but not necessarily, containing distinct questions and/or issues. Tabulation is performed on a Precinct level. For convenience, *Precinct Results* may be aggregated and published as unified *Election Results* data.

Election Encoding Subgroup (ElectionEncodingSubgroup) : The unique order q subgroup of the ElectionGroup, where q is specified in the CryptoGroupParameters structure contained in the CryptoElectionParameters structure of the BlankBallot.

Election Group (ElectionGroup) : The modular arithmetic group specified by the ElectionModulus (p) parameter of the CryptoGroupParameters

structure contained in the CryptoElectionParameters structure of the BlankBallot.

Election ID (**ElectionID**) **:** A UUID for elections.

Election Modulus (**ElectionModulus**) **:** The prime integer (p) which determines the ElectionGroup used for VotedBallot encryption. It is specified in the CryptoGroupParameters structure contained in the CryptoElectionParameters structure of the BlankBallot.

Election Public Key (**ElectionPublicKey**) **:** An element, h, of the ElectionEncodingSubgroup. The corresponding *private key* ($\log_g h$, where g is the ElectionSubgroupMember) is a secret cryptographically shared between a set of "election trustees" (AuthoritySet) via a Pedersen dealerless threshold scheme.

Election Results : A data structure containing all the data necessary to display the election results (tally) in an "official" human readable form. Most likely an XML structure containing general information such as *Election Name*, *Question Text* and *Answer Text* along with corresponding numerical tabulation results.

Election Subgroup Member (**ElectionSubgroupMember**) **:** A fixed element, g, of the ElectionGroup . Typically g will not be 1 (or $q-1$), in which case it generates the ElectionEncodingSubgroup; in particular, g then satisfies the following relationship with the ElectionSubgroupModulus, q:

$$|g| \;=\; q \tag{A.2}$$

However, g may be 1 in order to effectively neuter El Gamal encryption (each El Gamal pair will then consist of the number 1, and the unencrypted element); this may be useful for testing, or when voted ballots do not need to be secret.

Election Subgroup Modulus (**ElectionSubgroupModulus**) **:** The prime integer (q) which determines the ElectionEncodingSubgroup used for VotedBallot encryption. In addition to being prime, it must also be related to the ElectionModulus (p) by the relationships:

$$\begin{aligned} p-1 &= q\,r \\ (q\,,\,r) &= 1 \end{aligned} \tag{A.3}$$

It is specified in the CryptoGroupParameters structure contained in the CryptoElectionParameters structure of the BlankBallot.

ElGamal Pair (**ElGamalPair**) : A *pair* of *Modular Integers* (ModularInt).

$$\text{ElGamalPair} \doteq (X, Y)$$

Encrypted Data (**EncryptedData**) : A collection of data which can be decrypted, given a suitable GeneralPurposePrivateKey.

Encrypted Session Key (**EncryptedSessionKey**) : A random byte stream that has been encrypted with a GeneralPurposePublicKey. It is used to encrypt a message with a stream cipher. (The message is not encrypted directly with the GeneralPurposePublicKey because that would be too slow.)

Encryption Private Key (**EncryptionPrivateKey**) : A key which can be used for decryption.

Encryption Public Key (**EncryptionPublicKey**) : A key which can be used for encryption.

Error Structure (**ErrorStructure**) : Structure for encoding "unexpected" return conditions.

General Purpose Private Key (**GeneralPurposePrivateKey**) : A key which can be used for both decryption and signature generation.

General Purpose Public Key (**GeneralPurposePublicKey**) : A key which can be used for both encryption and signature validation.

Identification Information (**IdentInfo**) : An XML string containing identifying information about the owner/creator of a GeneralPurposePublicKey or GeneralPurposePrivateKey.

Initialization Vector (**InitializationVector**) : A short, pseudo-random byte stream that increases the security of a CipherText.

Key Generation Parameters (**KeyGenParameters**) : The parameters determining the number of Authorities (the KeyShareWidth, n) participating in Key Sharing and the number of these (the TabulationThreshold, t) who must cooperate in order to tabulate.

$$\text{KeyGenParameters} \doteq \left\{ \begin{array}{c} \text{CryptoGroupParameters} \\ \texttt{int}\ 1 \leq n\ (\text{KeyShareWidth}) \\ \texttt{int}\ 1 \leq t \leq n\ (\text{TabulationThreshold}) \end{array} \right\}$$

Key Share Commitment (**KeyShareCommitment**) : A modular integer with constraints based on the election crypto parameters

$$\text{KeyShareCommitment} \doteq C \in \text{ElectionEncodingSubgroup}$$

where

$$C = g^s$$

and

$$s = \text{SecretShare}$$

Key Share Width : A positive integer (n) that specifies the total number of Authorities officially participating in Key Sharing. KeyShareWidth is referred to as NumAuth in the DTD.

Modular Integer (**ModularInt**) : A BigInt

$$\text{ModularInt} \doteq \left\{ \text{BigInt}\ x \right\}$$

(Modulus is determined from context, not explicitly represented.)

Pair-wise Secret (**PairwiseSecret**) : Each Authority evaluates his polynomial at all of the AuthorityEvaluationPoint ID values, including his own.

$$\text{PairwiseSecret} \doteq (ID_i\,,\ ID_j\,,\ f_i(\beta_j))$$

The identifiers (ID_i, ID_j) designate the sender (ID_i) and the recipient (ID_j) authorities. Each ID is the *fingerprint* of the corresponding Authority object. (See Key Sharing for details.)

Pair-wise Secrets (PairwiseSecrets) : An XML string containing the PairwiseSecret from/to each authority.

Partially Decrypted Ballot Box (PartiallyDecryptedBallotBox) : A data structure containing all the decryption information necessary to both *tabulate* and *verify* with respect to a given CryptoElectionParameters (or, with respect to a given SignedBlankBallot structure, which contains a unique CryptoElectionParameters structure). The count is is only verifiable against the contained RawBallotBox component. Additional verification (see Shuffle Verification) is needed to certify that the count is derived properly from the official set of SignedVotedBallots.

$$\text{PartiallyDecryptedBallotBox} \doteq \left\{ \begin{array}{c} \text{RawBallotBox} \\ \text{AuthorityPartialDecrypts} \end{array} \right\}$$

Permutation (Permutation) : An XML structure with attribute Size=n and data containing a random ordering of the numbers 1 through n.

Precinct : Refers to an *"atomic"* sub-jurisdiction, its ballot and tabulation results. "Atomic" means that

1. All voters in a given Precinct must use (i.e. vote on) the same BlankBallot.
2. All ballots cast in a given Precinct must be tabulated together to produce a *single count*, or set of question/issue results. *Seperation of voters within a precinct into sub-categories is not allowed.* If a Precinct needs to be subdivided, it should be seperated into multiple Precincts before ballot casting begins (i.e. polls are opened).

Precinct ID (PrecinctID) : A UID for precincts. PrecinctIDs are required to be unique *within a fixed election*, but are not required to be universally unique. This allows PrecinctIDs to be reused over time by a jurisdiction.

Pre-Verification Code (PreVerificationCode) : A particular ElGamalPair returned to the Vote Client used to generate a VoteVerificationCode which is both voter specific and chosen answer specific.

Pre-Verification Codes (PreVerificationCodes) : A vector of $\nu \geq 0$ PreVerificationCodes:

$$\text{PreVerificationCodes} \doteq (\text{PreVerificationCode}_1, \ldots, \text{PreVerificationCode}_\nu)$$

Pre-Verification Code Box (PreVerificationCodeBox) : Each trustee generates a RawBallotBox with encrypted ElGamal pairs using his VoteVerificationKey and returns it inside a PreVerificationCodeBox structure.

Pre-Verification Code Boxes (PreVerificationCodeBoxes) : A collection of PreVerificationCodeBoxes from all trustees.

Question Reference (QuestionReference) : A small integer, which, in the context of a fixed election is uniquely associated with a specific question. Question references *must* be assigned sequentially from 1 to NumBallotQuestions (0 to NumBallotQuestions - 1) since the PreVerificationCodes (or VoteVerificationCodes) will be returned in this order.

Question Skeleton (QuestionSkeleton) :

$$\text{QuestionSkeleton} \doteq \left\{ \text{QuestionTextStructure}\ \text{AnswerSkeleton} \right\}$$

Question Text Structure (QuestionTextStructure) : A specification according to a yet to be determined data standard of the human readable data (display text, title, shorthand name, etc.) associated with a ballot question (perhaps complicated to support multiple languages, etc.).

Random Bits (RandomBits) : A collection of random, or pseudorandom bits.

Random Block (RandomBlock) : An array of RandomBits which may be generated by hashing certain seed values or may be generated by another method.

Random IJ State (RandomIJState) : An XML structure describing the current random numbers available. An attribute "SourceType" should be set to "PSEUDO" or "TRUE", depending on whether one is generating pseudorandom or true random numbers. Indices i and j indicate the index of the first bit in the sequence.

$$\text{RandomIJState} \doteq (\text{RandomSeedKey})$$

or

$$\text{RandomIJState} \doteq ((i_0, j_0, n_0, bits_0), \ldots, (i_m, j_m, n_m, bits_m))$$

Random Seed Key (RandomSeedKey): A short byte sequence used to seed the random-number generator.

Random Seed Keys (RandomSeedKeys): A sequence of RandomSeedKeys.

Random State (RandomState): An XML structure describing the current random numbers available. An attribute "SourceType" should be set to "PSEUDO" or "TRUE", depending on whether one is generating pseudorandom or true random numbers. In the first definition, an attribute "Index" is included to indicate the location of the pointer in the RandomBlock.

$$\text{RandomState} \doteq \left\{ \begin{array}{c} \text{RandomSeedKey} \\ \text{RandomBlock} \end{array} \right\}$$

or

$$\text{RandomState} \doteq (\text{NIL})$$

Raw Ballot Box (RawBallotBox): A vector of $B \geq 0$ *Raw Voted Ballots*

$$\text{RawBallotBox} \doteq (\text{RawVotedBallot}_1, \ldots, \text{RawVotedBallot}_B)$$

Raw Clear Text Ballot (RawClearTextBallot): A direct representation, in the context of a fixed BlankBallot, of a decrypted voted ballot – i.e. set of voter choices. Constructed as a vector of $\nu \geq 0$ AnswerMarks.

$$\text{ClearTextBallot} \doteq (\text{AnswerMark}_1, \ldots, \text{AnswerMark}_\nu)$$

Raw Clear Text Ballots (RawClearTextBallots): An XML structure containing one or more RawClearTextBallot.

Raw Voted Ballot (RawVotedBallot): A vector of $m \geq 1$ *El Gamal Pairs*:

$$\text{RawVotedBallot} \doteq (\text{ElGamalPair}_1, \ldots, \text{ElGamalPair}_m)$$

Return Code (**ReturnCode**) : int

Revealed Dictionary Secret (**RevealedDictionarySecret**) : A random number associated with a given QuestionReference in a BlankBallot.

Secret Coefficients (**SecretCoefficients**) : Cryptographic quantities specific to the Key Sharing protocol.

$$\text{SecretCoefficients} \doteq (\theta_1 , \ldots , \theta_t)$$

where

$$\theta \in \mathbf{Z}_q$$

Secret Share (**SecretShare**) : A modular integer (secret) with constraints based on the CryptoElectionParameters. The Secret Share, s for the authority with an AuthorityEvaluationPoint is and element of $\mathbf{Z}_q$ (where q is the ElectionSubgroupModulus) characterized by:

$$\text{SecretShare} \doteq s = f(\text{AuthorityEvaluationPoint}) =$$

$$\sum_{j=1}^{n} f_j(\text{AuthorityEvaluationPoint})$$

Seed Parameters (**SeedParameters**) : An XML string containing initial values for generating KeyGenParameters. The values indicate the number of Authority objects to be created, the threshold number of Authorities to be used in Tabulation, and a seed for generating random numbers.

Shuffle Validity Proof (**ShuffleValidityProof**) : A zero-knowledge proof of correctness for shuffle.

Simple Shuffle Validity Proof (**SimpleShuffleValidityProof**) : A zero-knowledge proof which is a part of the larger ShuffleValidityProof zero-knowledge proof.

Signature (**Signature**) : A digital signature as specified in a publicly accepted standard. Such a standard specifies with mathematical precision how the "validity" of any Signature relative to a particular data string (document) and GeneralPurposePublicKey, or Certificate, is determined.

Signed Ballot Box (**SignedBallotBox**) : The vector of Signed Voted Ballots that constitute input to tabulation. An authenticating signature (or vector of signatures) is appended for the purpose of "officially sealing" the ballot box. (The exact treatment of these signatures will be set as a matter of election policy.)

$$\text{SignedBallotBox} \doteq \left\{ \begin{array}{c} \text{ElectionID} \\ (\text{SignedVotedBallot}_1, \ldots, \text{SignedVotedBallot}_t) \\ (\text{Signature}_1, \ldots, \text{Signature}_I) \end{array} \right\}$$

Signed Blank Ballot (**SignedBlankBallot**) : A *Blank Ballot* with an arbitrary number of detached signatures.

$$\text{SignedBlankBallot} \doteq \left\{ \begin{array}{c} \text{BlankBallot} \\ (\text{Signature}_1, \ldots, \text{Signature}_t) \end{array} \right\}$$

Signed Document (**SignedDocument**) : An XML structure containing a hash of the original plaintext to be signed, and a Signature.

Signed Election Parameters (**SignedElectionParameters**) : The Crypto Election Prameters (CryptoElectionParameters), along with a (policy dependent) vector of authorizing signatures.

$$\text{SignedElectionParameters} \doteq \left\{ \begin{array}{c} \text{CryptoElectionParameters} \\ (\text{Signature}_1, \ldots, \text{Signature}_\rho) \end{array} \right\}$$

Signed Trustee Dictionary Commitments (**SignedTrusteeDictionaryCommitments**) : A TrusteeDictionaryCommitments with a Signature.

Signed Voted Ballot (**SignedVotedBallot**) : A *Voted Ballot* with detached (voter, or voting device) signature.

$$\text{SVB} \doteq \left\{ \begin{array}{c} \text{VotedBallot} \\ \text{Signature} \end{array} \right\}$$

Signed Voted Ballots (**SignedVotedBallots**) : An XML structure representing a set of zero or more SignedVotedBallots. The order of the elements is irrelevant.

Signing Private Key : A GeneralPurposePrivateKey used for signature generation.

Signing Public Key : A GeneralPurposePublicKey used for signature verification.

Tabulation Threshold : A positive integer (t) which specifies the number of CommittedAuthoritys who must cooperate in order to tabulate the SignedBallotBox. It is determined as a part of Key Sharing and can not be changed thereafter. It must, by nature, satisfy $1 \leq t \leq n$, where n is the KeyShareWidth parameter in the CommittedAuthoritySet structure of the CryptoTabulationParameters structure contained in the CryptoElectionParameters structure of the BlankBallot. TabulationThreshold is referred to as Threshold in the DTD.

Trustee Dictionary Commitments (TrusteeDictionaryCommitments) : A structure that represents all DictionaryCommitments for a *fixed* ElectionID, *fixed* PrecinctID, and *fixed* VoteVerificationTrustee. That is, a collection of $N_{BSN} \geq 1$ BSNDictionaryCommitments, where N_{BSN} is the number of BallotSequenceNumbers for the Precinct indicated by (ElectionID, PrecinctID) .

$$\text{TrusteeDictionaryCommitments} \doteq \left\{ \begin{array}{c} \text{Authority} \\ \text{BlankBallot} \\ (\text{BSNDictionaryCommitments}_1, \ldots, \text{BSNDictionaryCommitments}_{N_{BSN}}) \\ \text{Signature} \end{array} \right\}$$

Trustee Dictionary Commitments Set (TrusteeDictionaryCommitmentsSet) A vector of TrusteeDictionaryCommitments. That is, a collection of $N_{VVT} \geq 1$ TrusteeDictionaryCommitments, where N_{VVT} is the number of VoteVerificationTrustees for the Precinct indicated the ElectionID and PrecinctID in the TrusteeDictionaryCommitments's BlankBallot.

$$\text{TrusteeDictionaryCommitmentsSet} \doteq (\text{TrusteeDictionaryCommitments}_1, \ldots, \text{TrusteeDictionaryCommitments}_{N_{VVT}}$$

Trustee Revealed Dictionary Secrets (TrusteeRevealedDictionarySecrets) : A single trustee's collection of BSNRevealedDictionarySecrets for a given BlankBallot. The number of BSNRevealedDictionarySecrets is given by $N_{BSN} \geq 1$ where N_{BSN} is the number of unvoted BallotSequenceNumbers.

$$\text{TrusteeRevealedDictionarySecrets} \doteq \left\{ \begin{array}{c} \text{Authority} \\ \text{BlankBallot} \\ (\text{BSNRevealedDictionarySecrets}_1, \ldots, \text{BSNRevealedDictionarySecrets}_{N_{BSN}}) \end{array} \right\}$$

Trustee Revealed Dictionary Secrets Box (TrusteeRevealedDictionarySecretsBox) : A collection of all of the trustees' TrusteeRevealedDictionarySecrets.

UID (UID) : Unique identification number.

UUID (UUID) : Universally unique identification number.

Vote Receipt (VoteReceipt) : A VoteReceiptData object *signed* (by Vote Collection Agency)

$$\text{VoteReceipt} \doteq \left\{ \begin{array}{c} \text{VoteReceiptData} \\ \text{Signature} \end{array} \right\}$$

Vote Receipt Data (VoteReceiptData) : Data used for proof of voting. When signed (see VoteReceipt), can be used by voters to verify authenticity of their SignedVotedBallot in the election transcript (when it exists), and to mount a protest in case of discrepancy.

$$\text{VoteReceipt} \doteq \left\{ \begin{array}{c} \text{HASH}(\text{SignedVotedBallot}) \\ \text{BallotSequenceNumber} \\ (\text{PreVerificationCodes} \mid \text{VoteVerificationCodes}) \end{array} \right\}$$

Vote Verification Code (VoteVerificationCode) : A data string. Depending on the underlying protocol, it may be computed from a PreVerificationCode.

Vote Verification Codes (**VoteVerificationCodes**) : A vector of $\nu \geq 0$ VoteVerificationCodes.

$$\text{VoteVerificationCodes} \doteq (\text{VoteVerificationCode}_1, \ldots, \text{VoteVerificationCode}_\nu)$$

Vote Verification Dictionary (**VoteVerificationDictionary**) : Data that assigns a VoteVerificationCode to each AnswerReference on the BlankBallot for a fixed BallotSequenceNumber.

$$\text{VoteVerificationDictionary} \doteq \left\{ \begin{array}{c} \text{BallotSequenceNumber} \\ (\text{DictionaryQuestion}_1, \ldots, \text{DictionaryQuestion}_{NQ}) \end{array} \right\}$$

Vote Verification Dictionaries (**VoteVerificationDictionaries**) : A vector of VoteVerificationDictionarys.

Vote Verification Key (**VoteVerificationKey**) : A SecretKey used by an individual VoteVerificationTrustee to pseudo-randomly generate DictionaryCommitments.

Vote Verification Keys (**VoteVerificationKeys**) : A vector of VoteVerificationKeys.

Vote Verification Statement (**VoteVerificationStatement**) : A statement provided to the voter after voting which contains VoteVerificationCodes corresponding to his selections.

Vote Verification Statements (**VoteVerificationStatements**) : A collection of VoteVerificationStatements.

Vote Verification Trustee (**VoteVerificationTrustee**) : Currently a synonym for Authority.

Voted Ballot (VotedBallot) :

$$\text{VotedBallot} \doteq \left\{ \begin{array}{c} \text{HASH}(\text{BlankBallot}) \\ \text{ElectionID} \\ \text{BallotSequenceNumber} \\ \text{RawVotedBallot} \end{array} \right\}$$

Appendix B

VHTi Library Reference

B.1 auth.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef AUTH_H
#define AUTH_H
```



```
#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_authenticate (SignedVotedBallots signed_voted_ballots,
                   VoterRoll voter_roll,
                   BlankBallot blank_ballot,
                   RawBallotBox *raw_ballot_box);

#ifdef __cplusplus
}
#endif
#endif
```

B.2 check_comm.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef CHECK_COMM_H
#define CHECK_COMM_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_check_commitment (KeyGenParameters key_gen_parameters,
                       Authority authority,  // Authority whose
                       // commitment we want to check
                       BroadcastValues broadcast_values,
                       KeyShareCommitment keyshare_commitment,
                       CheckResults *check_comm_results);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.3 check_dictionary_secrets.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef CHECK_DICTIONARY_SECRETS_H
#define CHECK_DICTIONARY_SECRETS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_check_dictionary_secrets (TrusteeRevealedDictionarySecrets trustee_dict_secrets,
                               SignedTrusteeDictionaryCommitments trustee_dict_comm,
                               GeneralPurposePublicKey tsig_pubkey,
                               BlankBallot blank_ballot,
                               CheckResults *check_dictionary_secrets_result);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.4 check_partial_decrypt.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef CHECK_PARTIAL_DECRYPT_H
#define CHECK_PARTIAL_DECRYPT_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_check_partial_decrypt (RawBallotBox raw_ballot_box,
                            AuthorityPartialDecrypt auth_partial_decrypt,
                            SignedBlankBallot signed_blank_ballot,
                            GeneralPurposePublicKey ballot_signing_key,
                            CheckResults *check_partial_decrypt_result);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.5 check_pds_and_combine.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef CHECK_PARTIAL_DECRYPTS_AND_COMBINE_H
#define CHECK_PARTIAL_DECRYPTS_AND_COMBINE_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_check_partial_decrypts_and_combine (SignedBlankBallot signed_blank_ballot,
                                         GeneralPurposePublicKey ballot_signing_key,
                                         PartiallyDecryptedBallotBox pd_ballot_box,
                                         ClearTextBallots *clear_text_ballots,
                                         const char **combine_partial_decrypt_result)

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.6 check_shuffle.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef CHECK_SHUFFLE_H
#define CHECK_SHUFFLE_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_check_shuffle (RawBallotBox raw_ballot_box_before,
                    RawBallotBox raw_ballot_box_after,
                    SignedBlankBallot signed_blank_ballot,
                    GeneralPurposePublicKey ballot_signing_key,
                    ShuffleValidityProof shuffle_validity_proof,
                    CheckResults *check_answer_result);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.7 check_vc_pds_and_combine.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef CHECK_VCODE_PARTIAL_DECRYPT_AND_COMBINE_H
#define CHECK_VCODE_PARTIAL_DECRYPT_AND_COMBINE_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_check_vcode_partial_decrypts_and_combine
     (PreVerificationCodeBoxes pre_vcode_boxes,
     AuthorityPartialDecrypts auth_partial_decrypts,
     SignedVotedBallots signed_voted_ballots,
     SignedBlankBallot signed_blank_ballot,
     GeneralPurposePublicKey ballot_signing_key,
     int v_len,
```

```
    AlphabetEncoding v_alphabet,
    VoteVerificationStatements *verification_statements,
    const char **combine_partial_decrypt_result);

#ifdef __cplusplus
}
#endif
#endif
```

B.8 combine_dictionary_secrets.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef COMBINE_DICTIONARY_SECRETS_H
#define COMBINE_DICTIONARY_SECRETS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_combine_dictionary_secrets (TrusteeRevealedDictionarySecrets trustee_dict_secrets
                                 BlankBallot blank_ballot,
                                 int v_len,
                                 AlphabetEncoding v_alphabet,
                                 VoteVerificationDictionaries *verification_dictionari

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.9 crypt.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef __CRYPT_H
#define __CRYPT_H

#include <vhti/support.h>

#ifdef __cplusplus
extern "C" {
#endif

EXPORT_SYMBOL int
VHTI_encrypt(GeneralPurposePublicKey public_key,
             ConstCharStar plaintext,
             EncryptedData *encrypted_data);

EXPORT_SYMBOL int
VHTI_decrypt(GeneralPurposePrivateKey private_key,
             EncryptedData encrypted_data,
             ConstCharStar *plaintext);
```

```
EXPORT_SYMBOL int
VHTI_password_encrypt(Password password,
                      ConstCharStar plaintext,
                      EncryptedData *encrypted_data);

EXPORT_SYMBOL int
VHTI_password_decrypt(Password password,
                      EncryptedData encrypted_data,
                      ConstCharStar *plaintext);


#ifdef __cplusplus
} // extern "C"
#endif

#endif /* __CRYPT_H */
```

B.10 enc_ballot_pollsite.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef ENC_BALLOT_POLLSITE_H
#define ENC_BALLOT_POLLSITE_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_encrypt_ballot_pollsite (ClearTextBallot clear_text_ballot,
                              BlankBallot blank_ballot,
                              BallotSequenceNumber bsn,
                              RandomState random_state,
                              GeneralPurposePrivateKey ballot_signing_key,
                              SignedVotedBallot * signed_voted_ballot,
                              RandomState *random_state_out);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.11 error.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef __ERROR_H
#define __ERROR_H

#ifdef __cplusplus
extern "C" {
#endif

#include <vhti/export.h>

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
EXPORT_SYMBOL int
   VHTI_get_last_error(const char ** error_text);

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
EXPORT_SYMBOL int
```

```
   VHTI_get_last_error_number(void);

#ifdef __cplusplus
} // extern "C"
#endif

#endif /* __ERROR_H */
```

B.12 export.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef _EXPORT_H
#define _EXPORT_H

#if defined(WIN32)
#define EXPORT_SYMBOL __declspec(dllexport)
#else
#define EXPORT_SYMBOL
#endif

#if defined (_MFC_VER)
// Sneak this in here since it gets included into lots of stuff.  8/may/2003 ACB
// 4786: identifer was truncated in debug information
#pragma warning(disable: 4786)
#endif

#endif
```


B.13 gen_answer_mark.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_ANSWER_MARK_H
#define GEN_ANSWER_MARK_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_answer_mark (CryptoGroupParameters crypto_group_parameters,
                           AlphabetEncoding encoding,
                           RandomState random_state,
                           AnswerMark *answer_mark,
                           RandomState * ending_random_state);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.14 gen_blank_ballot.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_BLANK_BALLOT_H
#define GEN_BLANK_BALLOT_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_blank_ballot (BallotSkeleton ballot_skeleton,
                            CryptoElectionParameters cep,
                            AlphabetEncoding encoding,
                            BlankBallot * blank_ballot);

#ifdef __cplusplus
}
```

```
#endif
#endif
```

B.15 gen_broadcast.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_BROADCAST_H
#define GEN_BROADCAST_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_broadcast (KeyGenParameters key_gen_parameters,
                         SecretCoefficients secret_coefficients,
                         RandomState random_state,
                         BroadcastValue *broadcast_value,
                         RandomState *random_state_out);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.16 gen_bsns.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_BSNS_H
#define GEN_BSNS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_bsns (ElectionID eid,
    PrecinctID pid,
    int numAuthorized,
                    int numProvisional,
                    RandomState random_state,
                    BallotSequenceNumbers * authorized_bsns,
                    BallotSequenceNumbers * provisional_bsns,
```

```
                    RandomState *random_state_out);

#ifdef __cplusplus
}
#endif
#endif
```

B.17 gen_comm.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_COMM_H
#define GEN_COMM_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_commitment (KeyGenParameters key_gen_parameters,
                          Authority authority,  // This is the recipient
                                                // of the following secrets
                          BroadcastValues all_broadcast_values,
                          PairwiseSecrets pairwise_secrets,
                          SecretShare *secret_share,
                          KeyShareCommitment *keyshare_commitment);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.18 gen_dictionary_secrets.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_DICTIONARY_SECRETS_H
#define GEN_DICTIONARY_SECRETS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_dictionary_secrets (SignedVotedBallots signed_voted_ballots,
                                  Authority authority,
                                  GeneralPurposePrivateKey private_signature_key,
                                  VoteVerificationKey prn_seed,
                                  BlankBallot blank_ballot,
                                  BallotSequenceNumbers bsns,
                                  TrusteeRevealedDictionarySecrets *trustee_dict_secrets);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.19 gen_pre_verification_results.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_PRE_VV_RESULTS_H
#define GEN_PRE_VV_RESULTS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_pre_verification_results (VoteVerificationKey vv_key,
                                        SignedVotedBallots signed_voted_ballots,
                                        SignedBlankBallot signed_blank_ballot,
                                        GeneralPurposePublicKey ballot_signing_key,
                                        PreVerificationCodeBox * pre_vcode_box);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.20 gen_pubkey.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_PUBKEY_H
#define GEN_PUBKEY_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_public_key (KeyGenParameters key_gen_parameters,
                          BroadcastValues broadcast_values,
                          ElectionPublicKey *public_key);

#ifdef __cplusplus
}
#endif
```

```
#endif
```

B.21 gen_pv_results.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_PARTIAL_VV_RESULTS_H
#define GEN_PARTIAL_VV_RESULTS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_partial_verification_results
     (PreVerificationCodeBoxes pre_vcode_boxes,
      SignedBlankBallot signed_blank_ballot,
      GeneralPurposePublicKey ballot_signing_key,
      CommittedAuthority committed_authority,
      SecretShare secret_share,
      RandomState random_state,
```

```
      AuthorityPartialDecrypt *auth_partial_decrypt_of_verifications,
      RandomState *random_state_out);

#ifdef __cplusplus
}
#endif
#endif
```

B.22 gen_seccoeff.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_SECCOEFF_H
#define GEN_SECCOEFF_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_secret_coefficients (KeyGenParameters key_gen_parameters,
                                   Authority authority,
                                   RandomState random_state,
                                   SecretCoefficients *secret_coefficients,
                                   RandomState * random_state_out);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.23 gen_secrets.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_SECRETS_H
#define GEN_SECRETS_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_secret (KeyGenParameters key_gen_parameters,
                      Authority authority,
                      SecretCoefficients secret_coefficients,
                      PairwiseSecret *pairwise_secret);

#ifdef __cplusplus
}
```



```
#endif
#endif
```

B.24 gen_verification_code.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_VVCODE_H
#define GEN_VVCODE_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_verification_code (VoteVerificationKeys vv_keys,
                                 BlankBallot blank_ballot,
                                 BallotSequenceNumber bsn,
                                 AnswerReference answer_ref,
                                 int vv_len,
                                 AlphabetEncoding vv_alphabet,
                                 VoteVerificationCode * vv_code);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.25 gen_vote_receipt_data.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_VOTE_RECEIPT_DATA_H
#define GEN_VOTE_RECEIPT_DATA_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_vote_receipt_data (SignedVotedBallot signed_voted_ballot,
                                 VoteVerificationKeys vv_keys,
                                 BlankBallot blank_ballot,
                                 BallotSequenceNumber bsn,
                                 ClearTextBallot clear_text_ballot,
                                 int vv_len,
                                 AlphabetEncoding vv_alphabet,
```

```
                                    VoteReceiptData * vote_receipt_data);

#ifdef __cplusplus
}
#endif
#endif
```

B.26 gen_vvdict.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_VCDICT_H
#define GEN_VCDICT_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_vote_verification_dictionary (VoteVerificationKeys vv_keys,
                                            BlankBallot blank_ballot,
                                            BallotSequenceNumber bsn,
                                            int vv_len,
                                            AlphabetEncoding vv_alphabet,
                                            VoteVerificationDictionary * vv_dictionary);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.27 gen_vvdict_comm.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_VCDICT_COMM_H
#define GEN_VCDICT_COMM_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_vvdict_commits (Authority authority,
                              GeneralPurposePrivateKey private_signature_key,
                              VoteVerificationKey prn_seed,
                              BlankBallot blank_ballot,
                              BallotSequenceNumbers bsns,
                              SignedTrusteeDictionaryCommitments *trustee_dict_comm);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.28 gen_vvkey.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef GEN_VCKEY_H
#define GEN_VCKEY_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_vvk (RandomState random_state,
                   VoteVerificationKey * vv_key,
                   RandomState *random_state_out);

#ifdef __cplusplus
}
#endif
```

```
#endif
```

B.29 genkeys.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef __GENKEYS_H
#define __GENKEYS_H

#include <vhti/support.h>

#ifdef __cplusplus
extern "C" {
#endif

EXPORT_SYMBOL int
VHTI_generate_keys(IdentInfo ident_info,
                   GeneralPurposePrivateKey *pr,
                   GeneralPurposePublicKey  *pu);

#ifdef __cplusplus
} // extern "C"
#endif
```

```
#endif /* __GENKEYS_H */
```

B.30 keyshare_util.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef KEYSHARE_UTIL_H
#define KEYSHARE_UTIL_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_create_keygen_parameters(SeedParameters input_parameters,
                              RandomState random_state,
                              KeyGenParameters * output_parameters,
                              RandomState * random_state_out);

EXPORT_SYMBOL int
VHTI_create_authority(const char *authority_id,
```

```
                        const KeyGenParameters key_gen_parameters,
                        RandomState random_state,
                        const GeneralPurposePublicKey pu,
                        GeneralPurposePrivateKey *pr,
                        Authority * authority,
                        RandomState * random_state_out);

#ifdef __cplusplus
}
#endif
#endif
```

B.31 partial_decrypt.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef PARTIAL_DECRYPT_H
#define PARTIAL_DECRYPT_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_partial_decrypt (RawBallotBox raw_ballot_box,
                      SignedBlankBallot signed_blank_ballot,
                      GeneralPurposePublicKey ballot_signing_key,
                      CommittedAuthority committed_authority,
                      SecretShare secret_share,
                      RandomState random_state,
                      AuthorityPartialDecrypt *auth_partial_decrypt,
```

```
                    RandomState *random_state_out);

#ifdef __cplusplus
}
#endif
#endif
```

B.32 permutation.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef PERMUTATION_H
#define PERMUTATION_H

#include "vhti/support.h"
#include <openssl/bn.h>

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_get_permutation(RandomState random_state,
                     int nsize,
                     Permutation * perm,
                     RandomState * random_state_out);

#ifdef __cplusplus
```

```
}
#endif
#endif
```

B.33 random.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef RANDOM_H
#define RANDOM_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_generate_random_state(RandomSeedKey key,
                           RandomState * random_state);

EXPORT_SYMBOL int
VHTI_get_bits(RandomState random_state,
              int nbits,
              RandomState * return_random_state,
              RandomBits * bits);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.34 shuffle.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef SHUFFLE_H
#define SHUFFLE_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_shuffle (RawBallotBox raw_ballot_box_before,
              RandomState random_state,
              SignedBlankBallot signed_blank_ballot,
              GeneralPurposePublicKey ballot_signing_key,
              RawBallotBox *raw_ballot_box_after,
              RandomState *random_state_out,
              ShuffleValidityProof *shuffle_validity_proof);
```

```
#ifdef __cplusplus
}
#endif
#endif
```

B.35 sign.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef __SIGN_H
#define __SIGN_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C" {
#endif

EXPORT_SYMBOL int
VHTI_sign_xml (GeneralPurposePrivateKey private_key,
               ConstCharStar xml_plaintext,
               ConstCharStar *signed_xml);

   /* deprecated for signing XML.  Use the above instead. */
EXPORT_SYMBOL int
VHTI_sign(GeneralPurposePrivateKey private_key,
          ConstCharStar plaintext,
```

```
          Signature *signature);

EXPORT_SYMBOL int
VHTI_check_xml (GeneralPurposePublicKey public_key,
                ConstCharStar signed_xml,
                ConstCharStar expected_root_element_name,
                CheckResults *c_result,
                ConstCharStar *inner_xml);

   /* deprecated for checking signed XML.  Use the above instead. */
EXPORT_SYMBOL int
VHTI_verify_signature(GeneralPurposePublicKey public_key,
                      ConstCharStar plaintext,
                      Signature signature,
                      CheckResults *c_result);

#ifdef __cplusplus
} // extern "C"
#endif

#endif /* __SIGN_H */
```

B.36 sign_receipt.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef SIGN_RECEIPT_H
#define SIGN_RECEIPT_H

#include "vhti/support.h"

#ifdef __cplusplus
extern "C"
{
#endif

EXPORT_SYMBOL int
VHTI_sign_receipt (VoteReceiptData vote_receipt_data,
                   GeneralPurposePrivateKey receipt_signing_key,
                   SignedVoteReceiptData * vote_receipt);

#ifdef __cplusplus
}
#endif
```

```
#endif
```

B.37 support.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef SUPPORT_H
#define SUPPORT_H

#include "vhti/types.h"
#include "vhti/export.h"
#include "vhti/error.h"
#include <stdlib.h>              /* for size_t */

#if defined(_MSC_VER)
// 4786: identifer was truncated in debug information
#pragma warning(disable: 4786)
#endif

#ifdef __cplusplus

template <class kind>
class auto_freeing
{
```

```
public:
   auto_freeing(kind p = 0) : m_p(p) { }
   ~auto_freeing(void) { VHTI_free(m_p); }
   operator char * (void) { return (char *)m_p; }
   operator const kind (void) const { return m_p; }
   kind * operator & (void)
      {
         if (m_p)
         {
            VHTI_free (m_p);
            m_p = 0;
         }
         return & m_p;
      }
   void operator = (char * p)
      {
         if (m_p)
            VHTI_free (m_p);

         m_p = p;
      }
private:
   kind m_p;
};
#endif

#ifdef __cplusplus
extern "C"
{
#endif

// Public API

// For all return values, 0 is success; anything else is an error code
// which I haven't bothered defining.

// All 'out' 'char *' parameters must be freed with one of the
// 'VHTI_free' functions.

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
EXPORT_SYMBOL char *
VHTI_alloc(size_t num_bytes);

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
```



```
EXPORT_SYMBOL int
VHTI_free (const char *thing);

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
// As above, but does not zero the freed block.
EXPORT_SYMBOL int
VHTI_free_without_clearing (const char *thing);

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
EXPORT_SYMBOL char *
VHTI_dup (const char *thing);

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
// Frees all allocations that haven't already been freed with
// 'VHTI_free'.
EXPORT_SYMBOL int
VHTI_free_all();

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
// You'd expect this function to be declared in types.h, since it is
// defined in types.cpp.  However, the perl code that automatically
// generates the perl wrapper makes a point of not reading types.h.
// Therefore we declare it here.
EXPORT_SYMBOL int
VHTI_set_DTD_location (const char *path);

/* This function does not follow the normal VHTI conventions, and its
 * documentation is not auto-generated.  If you change this prototype,
 * please also change the corresponding LaTeX documentation. */
/* 0 means "valid" */
EXPORT_SYMBOL int
VHTI_validate (const char * datatype, const char * data);


#ifdef __cplusplus
}
#endif
#endif
```

B.38 types.h

```
/*  */
/* This material is subject to the VoteHere Source Code Evaluation */
/* License Agreement ("Agreement").  Possession and/or use of this */
/* material indicates your acceptance of this Agreement in its entirety. */
/* Copies of the Agreement may be found at www.votehere.net. */
/*  */
/* Copyright 2004 VoteHere, Inc.  All Rights Reserved */
/*  */
/* You may not download this Software if you are located in any country */
/* (or are a national of a country) subject to a general U.S. or */
/* U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, */
/* Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States */
/* (each a "Prohibited Country") or are otherwise denied export */
/* privileges from the United States or Canada ("Denied Person"). */
/* Further, you may not transfer or re-export the Software to any such */
/* country or Denied Person without a license or authorization from the */
/* U.S. government.  By downloading the Software, you represent and */
/* warrant that you are not a Denied Person, are not located in or a */
/* national of a Prohibited Country, and will not export or re-export to */
/* any Prohibited Country or Denied Party. */
// Copyright 2003 VoteHere Inc. All Rights Reserved.

#ifndef VHTI_TYPES_H
#define VHTI_TYPES_H

#ifdef __cplusplus
extern "C"
{
#endif

#include "export.h"

#include "./auto-macros.h"

/* Some handy strings for returning CheckResults. */
#define CHECK_SUCCESS_TEXT "0:Success"
#define CHECK_FAILURE_TEXT "1:Failure"
#define CHECK_SUCCESS_XML "<CheckResults>" CHECK_SUCCESS_TEXT "</CheckResults>"
#define CHECK_FAILURE_XML "<CheckResults>" CHECK_FAILURE_TEXT "</CheckResults>"
```



```
#include "./auto-typedefs.h"

#if 0
  /* this #if 0 is because we want this block visible to the document */
  /* preprocessor but not to the C compiler. */
  typedef int int;
  typedef enum enum;
#endif

#ifdef __cplusplus
}
#endif
#endif
```

Appendix C

VHTi XML DTDs

C.1 vhti.dtd

```
<!--  -->
<!-- This material is subject to the VoteHere Source Code Evaluation -->
<!-- License Agreement ("Agreement").  Possession and/or use of this -->
<!-- material indicates your acceptance of this Agreement in its entirety. -->
<!-- Copies of the Agreement may be found at www.votehere.net. -->
<!--  -->
<!-- Copyright 2004 VoteHere, Inc.  All Rights Reserved -->
<!--  -->
<!-- You may not download this Software if you are located in any country -->
<!-- (or are a national of a country) subject to a general U.S. or -->
<!-- U.N. embargo or are deemed to be a terrorist country (i.e., Cuba, -->
<!-- Iran, Iraq, Libya, North Korea, Sudan and Syria) by the United States -->
<!-- (each a "Prohibited Country") or are otherwise denied export -->
<!-- privileges from the United States or Canada ("Denied Person"). -->
<!-- Further, you may not transfer or re-export the Software to any such -->
<!-- country or Denied Person without a license or authorization from the -->
<!-- U.S. government.  By downloading the Software, you represent and -->
<!-- warrant that you are not a Denied Person, are not located in or a -->
<!-- national of a Prohibited Country, and will not export or re-export to -->
<!-- any Prohibited Country or Denied Party. -->
<!-- Questions about VoteHere's VHTi DTD may be addressed to: vhtifeedback@votehere.co
<!-- Alphabet Encoding -->
```



```
<!ELEMENT AlphabetEncoding (#PCDATA)>


<!-- Answer Partial Decrypt -->
<!ELEMENT AnswerPartialDecrypt (XValue, ConstantVal, FunctionVal)>
<!ELEMENT ConstantVal (#PCDATA)>
<!ATTLIST ConstantVal
Encoding CDATA #REQUIRED>
<!ELEMENT FunctionVal (#PCDATA)>
<!ATTLIST FunctionVal
Encoding CDATA #REQUIRED>

<!-- Answer Reference -->
<!ELEMENT AnswerReference (#PCDATA)>


<!-- Answer Skeleton -->
<!ELEMENT AnswerSkeleton (AnswerTextStructure?)>
<!ATTLIST AnswerSkeleton
AnswerReference CDATA #IMPLIED >

<!-- Authority -->
<!ELEMENT Authority (Certificate, AuthorityEvaluationPoint)>
<!ATTLIST Authority
AuthFingerprint CDATA #REQUIRED>
<!ELEMENT AuthorityEvaluationPoint (#PCDATA)>
<!ATTLIST AuthorityEvaluationPoint
Encoding CDATA #REQUIRED>


<!-- Authority Partial Decrypt -->
<!ELEMENT AuthorityPartialDecrypt (CommittedAuthority, BallotBoxPartialDecrypt)>


<!-- Authority Partial Decrypts -->
<!ELEMENT AuthorityPartialDecrypts (AuthorityPartialDecrypt+)>


<!-- Authority Set -->
<!ELEMENT AuthoritySet (Authority+)>


<!-- Ballot Answer -->
<!ELEMENT BallotAnswer (AnswerTextStructure, AnswerMark)>
<!ATTLIST BallotAnswer
AnswerReference CDATA #REQUIRED>
<!ELEMENT AnswerTextStructure (#PCDATA)>
<!ELEMENT AnswerMark (#PCDATA)>
```



```
<!ATTLIST AnswerMark
Encoding CDATA #REQUIRED>


<!-- Ballot Partial Decrypt -->
<!ELEMENT BallotPartialDecrypt (AnswerPartialDecrypt+)>


<!-- Ballot Box Partial Decrypt -->
<!ELEMENT BallotBoxPartialDecrypt (BallotPartialDecrypt*)>


<!-- Ballot Question -->
<!ELEMENT BallotQuestion (QuestionTextStructure, BallotAnswer+)>
<!ATTLIST BallotQuestion
QuestionReference CDATA #REQUIRED>


<!-- Ballot Questions -->
<!ELEMENT BallotQuestions (BallotQuestion+)>


<!-- Ballot Sequence Number -->
<!ELEMENT BallotSequenceNumber (#PCDATA)>
<!ATTLIST BallotSequenceNumber
Encoding CDATA #REQUIRED>


<!-- Ballot Sequence Numbers -->
<!ELEMENT BallotSequenceNumbers (BallotSequenceNumber+)>


<!-- Ballot Skeleton -->
<!ELEMENT BallotSkeleton (ElectionID, PrecinctID, QuestionSkeleton+, BallotTextStructu


<!-- Blank Ballot -->
<!ELEMENT BlankBallot (ElectionID, PrecinctID, CryptoElectionParameters, BallotQuestio
<!ELEMENT BallotTextStructure (#PCDATA)>


<!-- Broadcast Values -->
<!ELEMENT BroadcastValues (BroadcastValue+)>


<!-- Broadcast Value -->
<!ELEMENT BroadcastValue (BigTheta+, Omega, Something)>
<!ATTLIST BroadcastValue
```



```
AuthFingerprint CDATA #REQUIRED>
<!ELEMENT BigTheta (#PCDATA)>
<!ATTLIST BigTheta
Order CDATA #REQUIRED
Encoding CDATA #REQUIRED>
<!ELEMENT Omega (#PCDATA)>
<!ATTLIST Omega
Encoding CDATA #REQUIRED>
<!ELEMENT Something (#PCDATA)>
<!ATTLIST Something
Encoding CDATA #REQUIRED>


<!-- BSN Dictionary Commitments -->
<!ELEMENT BSNDictionaryCommitments (BallotSequenceNumber, DictionaryCommitment+)>


<!-- BSN Revealed Dictionary Secrets -->
<!ELEMENT BSNRevealedDictionarySecrets (BallotSequenceNumber, RevealedDictionarySecret+)>


<!-- Certificate -->
<!-- This probably needs more information, such as a human-readable -->
<!-- string indicating who made the signatures -->
<!ELEMENT Certificate (GeneralPurposePublicKey, Signature*)>


<!-- Check Results -->
<!ELEMENT CheckResults (#PCDATA)>


<!-- Clear Text Ballot -->
<!-- Secret --> <!ELEMENT ClearTextBallot (AnswerReference+)>


<!-- Clear Text Ballots -->
<!ELEMENT ClearTextBallots (ClearTextBallot+)>


<!-- Committed Authority -->
<!ELEMENT CommittedAuthority (Authority, KeyShareCommitment)>


<!-- Committed Authority Set -->
<!ELEMENT CommittedAuthoritySet (CommittedAuthority+)>


<!-- Crypto Election Parameters -->
```



```
<!ELEMENT CryptoElectionParameters (CryptoGroupParameters,
                                    CryptoTabulationParameters)>


<!-- Crypto Group Parameters -->
<!ELEMENT CryptoGroupParameters (ElectionModulus, ElectionSubgroupModulus, ElectionSub
<!ELEMENT ElectionModulus (#PCDATA)>
<!ATTLIST ElectionModulus
Encoding CDATA #REQUIRED>
<!ELEMENT ElectionSubgroupModulus (#PCDATA)>
<!ATTLIST ElectionSubgroupModulus
Encoding CDATA #REQUIRED>
<!ELEMENT ElectionSubgroupMember (#PCDATA)>
<!ATTLIST ElectionSubgroupMember
Encoding CDATA #REQUIRED>


<!-- Crypto Tabulation Parameters -->
<!ELEMENT CryptoTabulationParameters (ElectionPublicKey,
                                     NumAuth, Threshold, KeyShareCommitment+)>
<!ELEMENT ElectionPublicKey (#PCDATA)>
<!ATTLIST ElectionPublicKey
Encoding CDATA #REQUIRED>

<!-- NumAuth is the number of Authority entities -->
<!-- participating in the election               -->
<!ELEMENT NumAuth (#PCDATA)>
<!-- Threshold is the number of Authorities      -->
<!-- participating in keysharing                 -->
<!ELEMENT Threshold (#PCDATA)>

<!-- Dictionary Commitment -->
<!ELEMENT DictionaryCommitment (#PCDATA)>
<!ATTLIST DictionaryCommitment
Encoding CDATA #REQUIRED>


<!-- Dictionary Question  -->
<!ELEMENT DictionaryQuestion (DictionaryVerificationCode+)>
<!ATTLIST DictionaryQuestion
QuestionReference CDATA #REQUIRED>


<!-- Dictionary Verification Code  -->
<!ELEMENT DictionaryVerificationCode (#PCDATA)>
<!ATTLIST DictionaryVerificationCode
AnswerReference CDATA #REQUIRED
Encoding CDATA #REQUIRED>
```



```
<!-- Election ID -->
<!ELEMENT ElectionID (#PCDATA)>


<!-- ElGamal Pair -->
<!ELEMENT ElGamalPair (XValue, YValue)>
<!ELEMENT XValue (#PCDATA)>
<!ATTLIST XValue
Encoding CDATA #REQUIRED>
<!ELEMENT YValue (#PCDATA)>
<!ATTLIST YValue
Encoding CDATA #REQUIRED>

<!-- Encrypted Data -->
<!ELEMENT EncryptedData (InitializationVector, EncryptedSessionKey, CipherText)>
<!ELEMENT InitializationVector (#PCDATA)>
<!ELEMENT EncryptedSessionKey (#PCDATA)>
<!ELEMENT CipherText (#PCDATA)>


<!-- EncryptionPrivateKey -->
<!-- Secret --> <!ELEMENT EncryptionPrivateKey (#PCDATA)>


<!-- EncryptionPublicKey -->
<!ELEMENT EncryptionPublicKey (#PCDATA)>


<!-- Error Structure -->
<!ELEMENT ErrorStructure (#PCDATA)>


<!-- General Purpose Private Key -->
<!-- Secret --> <!ELEMENT GeneralPurposePrivateKey (IdentInfo, SigningPrivateKey, EncryptionPrivate


<!-- General Purpose Public Key -->
<!ELEMENT GeneralPurposePublicKey  (IdentInfo, SigningPublicKey , EncryptionPublicKey )>


<!-- Identification Information -->
<!ELEMENT IdentInfo (#PCDATA)>


<!-- Key Generation Parameters -->
<!ELEMENT KeyGenParameters (CryptoGroupParameters, NumAuth, Threshold)>
```



```
<!-- Key Share Commitment -->
<!ELEMENT KeyShareCommitment (#PCDATA)>
<!ATTLIST KeyShareCommitment
AuthFingerprint CDATA #REQUIRED
Encoding CDATA #REQUIRED>

<!-- Modular Integer -->
<!ELEMENT ModularInt (#PCDATA)>


<!--Pairwise Secrets -->
<!-- Secret --> <!ELEMENT PairwiseSecrets (PairwiseSecret+)>


<!-- Pairwise Secret -->
<!-- Secret --> <!ELEMENT PairwiseSecret (#PCDATA)>
<!ATTLIST PairwiseSecret
FromID CDATA #REQUIRED
ToID CDATA #REQUIRED
Encoding CDATA #REQUIRED>


<!-- Password -->
<!-- A secret key used for data encryption -->
<!ELEMENT Password (#PCDATA)>


<!-- Password Encrypted Data -->
<!-- Data whose content is protected by a password -->
<!ELEMENT PasswordEncryptedData (CipherText)>


<!-- Partially Decrypted Ballot Box -->
<!ELEMENT PartiallyDecryptedBallotBox (RawBallotBox, AuthorityPartialDecrypts)>


<!-- Permutation -->
<!ELEMENT Permutation (#PCDATA)>
<!ATTLIST Permutation
Size CDATA #REQUIRED>


<!-- PrecinctID -->
<!ELEMENT PrecinctID (#PCDATA)>
```



```
<!-- Pre-Verification Code -->
<!ELEMENT PreVerificationCode (#PCDATA)>


<!-- Pre-Verification Codes -->
<!ELEMENT PreVerificationCodes (PreVerificationCode+)>


<!-- Pre Verification Code Box -->
<!ELEMENT PreVerificationCodeBox (RawBallotBox)>


<!-- Pre Verification Code Boxes -->
<!ELEMENT PreVerificationCodeBoxes (PreVerificationCodeBox+)>


<!-- Question Reference -->
<!ELEMENT QuestionReference (#PCDATA)>


<!-- Question Skeleton -->
<!ELEMENT QuestionSkeleton (QuestionTextStructure?, AnswerSkeleton*)>
<!ATTLIST QuestionSkeleton
QuestionReference CDATA #IMPLIED
NumAns CDATA #IMPLIED>


<!-- Question Text Structure -->
<!ELEMENT QuestionTextStructure (#PCDATA)>


<!-- Random Bits -->
<!ELEMENT RandomBits (#PCDATA)>
<!ATTLIST RandomBits
Encoding CDATA #REQUIRED>


<!-- Random Block -->
<!-- Secret --> <!ELEMENT RandomBlock (#PCDATA)>
<!ATTLIST RandomBlock
Encoding CDATA #REQUIRED>


<!-- Random IJ State -->
<!-- Secret --> <!ELEMENT RandomIJState (RandomSeedKey)>
<!ATTLIST RandomIJState
SourceType CDATA #REQUIRED>
```



```
<!-- RandomSeedKey -->
<!-- Secret --> <!ELEMENT RandomSeedKey (#PCDATA)>


<!-- RandomSeedKeys -->
<!-- Secret --> <!ELEMENT RandomSeedKeys (RandomSeedKey+)>


<!-- Random State -->
<!-- Secret --> <!ELEMENT RandomState (RandomSeedKey, RandomBlock)>
<!ATTLIST RandomState
SourceType CDATA #REQUIRED>


<!-- Raw Ballot Box -->
<!ELEMENT  RawBallotBox (RawVotedBallot)* >


<!-- Raw Clear Text Ballot -->
<!ELEMENT RawClearTextBallot (AnswerMark+)>


<!-- Raw Clear Text Ballots -->
<!ELEMENT RawClearTextBallots (RawClearTextBallot+)>


<!-- Raw Voted Ballot -->
<!ELEMENT RawVotedBallot (ElGamalPair)+ >


<!-- Results -->
<!-- The specification is left to the application. -->
<!ELEMENT Results (#PCDATA)>


<!-- Return Code -->
<!ELEMENT ReturnCode (#PCDATA)>


<!-- Revealed Dictionary Secret -->
<!ELEMENT RevealedDictionarySecret (#PCDATA)>
<!ATTLIST RevealedDictionarySecret
Encoding CDATA #REQUIRED>


<!-- Secret Coefficients -->
<!-- Secret --> <!ELEMENT SecretCoefficients (SmallTheta+)>
```



```
<!ATTLIST SecretCoefficients
AuthFingerprint CDATA #REQUIRED>
<!-- Secret --> <!ELEMENT SmallTheta (#PCDATA)>
<!ATTLIST SmallTheta
Order CDATA #REQUIRED
Encoding CDATA #REQUIRED>

<!-- Secret Share -->
<!-- Secret --> <!ELEMENT SecretShare (#PCDATA)>
<!ATTLIST SecretShare
Encoding CDATA #REQUIRED>


<!-- Seed Parameters -->
<!ELEMENT SeedParameters (NumAuth, Threshold)>


<!-- Shuffle Validity Proofs -->
<!ELEMENT ShuffleValidityProofs (ShuffleValidityProof+)>


<!-- Shuffle Validity Proof -->
<!ELEMENT ShuffleValidityProof (U_Values, W_Values, A_Values, C_Values,
                                Lambda_1, Lambda_2, sigma_values, tau,
                                SimpleShuffleValidityProof)>
<!ELEMENT U_Values (UValue*)>
<!ELEMENT UValue (#PCDATA)>
<!ATTLIST UValue
Encoding CDATA #REQUIRED>
<!ELEMENT W_Values (WValue*)>
<!ELEMENT WValue (#PCDATA)>
<!ATTLIST WValue
Encoding CDATA #REQUIRED>
<!ELEMENT A_Values (AValue*)>
<!ELEMENT AValue (#PCDATA)>
<!ATTLIST AValue
Encoding CDATA #REQUIRED>
<!ELEMENT C_Values (CValue*)>
<!ELEMENT CValue (#PCDATA)>
<!ATTLIST CValue
Encoding CDATA #REQUIRED>
<!ELEMENT Lambda_1 (#PCDATA)>
<!ATTLIST Lambda_1
Encoding CDATA #REQUIRED>
<!ELEMENT Lambda_2 (#PCDATA)>
<!ATTLIST Lambda_2
Encoding CDATA #REQUIRED>
<!ELEMENT sigma_values (sigma*)>
```



```
<!ELEMENT sigma (#PCDATA)>
<!ATTLIST sigma
Encoding CDATA #REQUIRED>
<!ELEMENT tau (#PCDATA)>
<!ATTLIST tau
Encoding CDATA #REQUIRED>

<!-- Simple Shuffle Validity Proof Original
// <!ELEMENT SimpleShuffleValidityProof (BigGamma, Xbar_kp1, Ybar_kp1, Theta, alpha_va
// <!ELEMENT BigGamma (#PCDATA)>
// <!ATTLIST BigGamma
// Encoding CDATA #REQUIRED>
// <!ELEMENT Xbar_kp1 (#PCDATA)>
// <!ATTLIST Xbar_kp1
// Encoding CDATA #REQUIRED>
// <!ELEMENT Ybar_kp1 (#PCDATA)>
// <!ATTLIST Ybar_kp1
// Encoding CDATA #REQUIRED>
// <!ELEMENT Theta (#PCDATA)>
// <!ATTLIST Theta
// Encoding CDATA #REQUIRED>
// <!ELEMENT alpha_values (alpha+)>
// <!ELEMENT alpha (#PCDATA)>
// <!ATTLIST alpha
// Encoding CDATA #REQUIRED -->

<!-- Simple Shuffle Validity Proof Short Version
// <!ELEMENT SimpleShuffleValidityProof (GValue, BigGamma, c_hash, alpha_values)>
// <!ELEMENT GValue (#PCDATA)>
// <!ATTLIST GValue
// Encoding CDATA #REQUIRED>
// <!ELEMENT c_hash (#PCDATA)>
// <!ATTLIST c_hash
// Encoding CDATA #REQUIRED -->

<!-- Simple Shuffle Validity Proof Long Version -->
<!ELEMENT SimpleShuffleValidityProof (GValue, BigGamma, Theta_values, alpha_values)>
<!ELEMENT GValue (#PCDATA)>
<!ATTLIST GValue
Encoding CDATA #REQUIRED>
<!ELEMENT BigGamma (#PCDATA)>
<!ATTLIST BigGamma
Encoding CDATA #REQUIRED>
<!ELEMENT Theta_values (Theta_value*)>
<!ELEMENT Theta_value (#PCDATA)>
<!ATTLIST Theta_value
Encoding CDATA #REQUIRED>
<!ELEMENT alpha_values (alpha*)>
```



```
<!ELEMENT alpha (#PCDATA)>
<!ATTLIST alpha
Encoding CDATA #REQUIRED>

<!-- Simple Shuffle Validity Proof Optimized
// <!ELEMENT SimpleShuffleValidityProof (GValue, BigGamma, BigOmega, f_values, Theta_values, Theta_
// <!ELEMENT BigOmega (#PCDATA)>
// <!ATTLIST BigOmega
// Encoding CDATA #REQUIRED>
// <!ELEMENT Theta_values (Theta_value+)>
// <!ELEMENT Theta_value (#PCDATA)>
// <!ATTLIST Theta_value
// Encoding CDATA #REQUIRED>
// <!ELEMENT f_values (f_value+)>
// <!ELEMENT f_value (#PCDATA)>
// <!ATTLIST f_value
// Encoding CDATA #REQUIRED>
// <!ELEMENT Theta_kp1 (#PCDATA)>
// <!ATTLIST Theta_kp1
// Encoding CDATA #REQUIRED>
// <!ELEMENT Theta_2k (#PCDATA)>
// <!ATTLIST Theta_2k
// Encoding CDATA #REQUIRED -->

<!-- Signature -->
<!ELEMENT Signature (#PCDATA)>


<!-- SignedData -->
<!ELEMENT SignedData (#PCDATA)>


<!-- Signed Ballot Box -->
<!ELEMENT SignedBallotBox (#PCDATA)>


<!-- Signed Blank Ballot -->
<!ELEMENT SignedBlankBallot (SignedData, Signature+)>


<!-- Signed Election Parameters -->
<!ELEMENT SignedElectionParameters (CryptoElectionParameters,
                                    KeyShareCommitment+, Signature+)>


<!-- Signed Vote Receipt Data -->
<!ELEMENT SignedVoteReceiptData (SignedData, Signature)>
```



```
<!-- Signed Voted Ballot -->
<!ELEMENT SignedVotedBallot (SignedData, Signature)>


<!-- Signed Voted Ballots -->
<!ELEMENT SignedVotedBallots (SignedVotedBallot*)>


<!-- SigningPrivateKey -->
<!ELEMENT SigningPrivateKey (#PCDATA)>


<!-- SigningPublicKey -->
<!ELEMENT SigningPublicKey (#PCDATA)>


<!-- Trustee Dictionary Commitments -->
<!ELEMENT TrusteeDictionaryCommitments (Authority, BlankBallot,
                                        BSNDictionaryCommitments+)>


<!-- Trustee Dictionary Commitments Set -->
<!ELEMENT TrusteeDictionaryCommitmentsSet (TrusteeDictionaryCommitments+)>


<!-- Trustee Revealed Dictionary Secrets -->
<!ELEMENT TrusteeRevealedDictionarySecrets (Authority, BlankBallot,
                                      BSNRevealedDictionarySecrets+)>


<!-- Trustee Revealed Dictionary Secrets Box -->
<!ELEMENT TrusteeRevealedDictionarySecretsBox (TrusteeRevealedDictionarySecrets*)>


<!-- Signed Trustee Dictionary Commitments -->
<!ELEMENT SignedTrusteeDictionaryCommitments (SignedData, Signature)>


<!-- Vote Receipt Data -->
<!ELEMENT VoteReceiptData (BallotSequenceNumber, VoteVerificationCodes)>
<!ATTLIST VoteReceiptData
SVBHash CDATA #REQUIRED>


<!-- Vote Verification Code -->
<!ELEMENT VoteVerificationCode (#PCDATA)>
<!ATTLIST VoteVerificationCode
```



```
Encoding CDATA #REQUIRED
QuestionReference CDATA #REQUIRED>


<!-- Vote Verification Codes -->
<!ELEMENT VoteVerificationCodes (VoteVerificationCode+)>


<!-- Vote Verification Dictionary  -->
<!ELEMENT VoteVerificationDictionary (BallotSequenceNumber, DictionaryQuestion+)>


<!-- Vote Verification Dictionaries  -->
<!ELEMENT VoteVerificationDictionaries (VoteVerificationDictionary*)>


<!-- Vote Verification Key -->
<!ELEMENT VoteVerificationKey (#PCDATA)>


<!-- Vote Verification Keys -->
<!ELEMENT VoteVerificationKeys (VoteVerificationKey+)>


<!-- Vote Verification Statement -->
<!ELEMENT VoteVerificationStatement (BallotSequenceNumber, VoteVerificationCode+)>


<!-- Vote Verification Statements -->
<!ELEMENT VoteVerificationStatements (VoteVerificationStatement*)>


<!-- Voted Ballot -->
<!ELEMENT VotedBallot (ElectionID, BallotSequenceNumber, RawVotedBallot)>
<!ATTLIST VotedBallot
BBHash CDATA #REQUIRED>


<!-- Voter Roll -->
<!ELEMENT VoterRoll (Certificate*)>
```
