Election Verification Technology

Terminology

September 15, 2003

Chapter 1

Common Terms

Across the election industry, the meaning of terms tends to vary considerably since elections involve a multitude of disciplines ranging from election administration, political science, information security, physical security, and computer science. The intent here is to be clear in the meaning of common terms as they are used throughout the VHTi documentation.

For more detail, note that these **Common Terms** will often reference similar terms in the **Data Definitions** section.

Answer : Text representing a choice to a Question on the BlankBallot. Also, a corresponding code, or AnswerMark, is assigned to each *Answer* for use in the cryptographic protocols.

See also BallotAnswer and AnswerMark.

Ballot : A series of Questions and Answers used to Capture voter choices. Requirements need not prescribe specific properties but shall provide a formal ballot specification satisfying the following:

1. The specification shall be sufficiently detailed to allow all ElectionObservers to determine, with perfect consistency, whether some data does, or does not, constitute a ballot.
2. The specification shall be publicly available.

See also BlankBallot, VotedBallot, and SignedVotedBallot.

Ballot Box : A collection of Ballots.
See also SealedBallotBox.

Ballot Definition : A series of Questions and Answers.
See also BlankBallot and Ballot.

Ballot Presentation : The presentation of the Ballot to the voter in the voting booth.

Ballot Receipt : See VerificationReceipt.

Ballot Shuffling : See Shuffle.

Ballot Storage Device : A device used to store ballot data such as memory card, smartcard, hard drive, floppy, etc.

BSN : See BallotSequenceNumber.

Canvass : An internal audit, usually required by applicable law, to ensure the accuracy of the Results.

Capture : See VoteCapture.

Casting : See VoteCasting.

Choice : See VoterIntent.

Codebook : See VoteVerificationCodebook.

Codebook Commitments : Published before the election to ensure that VerificationCodes cannot be later changed without detection.
See also CodebookCommitment.

Codebook Correspondence : The correspondence between VerificationCodes and Answers. This correspondence is different for each BallotSequenceNumber and committed before the election.
See also VerificationCodebook and CodebookCommitments.

Codebook Key : Jointly generated by the Trustees prior to the election (see PedersenSecretSharingProtocol), the *Codebook Key* tells the DRE how to generate the individual VerificationCodebooks from each BallotSequenceNumber.
See also CodebookCorrespondence.

Counting : See VoteCounting.

Decrypted Ballot : Plain text ballot used in Tabulation after the EncryptedBallot is Shuffled and decrypted by the Trustees.

Decryption Key : Used to decrypt the EncryptedBallots after they are Shuffled in the BallotBox.
See also TrusteePrivateKey, KeyShare, and PedersenSecretSharingProtocol.

Distribution Medium : Used for data storage such as memory card, smartcard, hard drive, floppy, etc.

DRE : Direct Recording Equipment, also known as electronic voting machine, vote client, or touch-screen voting machine.

DRE Private Key : Used by the DRE to sign EncryptedBallots and VerificationReceipts.

Election Auditor : See ElectionObserver.

Election Configuration : The entire configuration for a given election, including ElectionParameters, CodebookKey, DREPivateKey, BallotSequenceNumbers, SignedBlankBallots, etc.

Election Management Computer : The computer used by the ElectionOfficials to generate and publish the ElectionConfiguration and tabulate and publish the Results. Typically, the *Election Management Computer* involves many computers.

Election Observer : Any individual or organization that scrutinizes the election.

Election Official : Personnel officially empowered to administer the election. Typically, the *Election Official* is one of the Trustees.

Election Parameters : Cryptographic parameters that must be generated, accepted, and published by the election Trustees before the election begins, in order to encrypt ballots, shuffle and tabulate the election results.
See also CryptoElectionParameters.

Election Transcript : After the election is tabulated, the Trustees generate the *Election Transcript* report consisting of all data required for ResultsVerification. This includes, at least, the ElectionConfiguration, EncryptedBallots, ShuffleProofs, Results, and the VerificationStatements for each BallotSequenceNumber used in the Tabulation. Along with the VoterRoll, the *Election Transcript* is used in the official election Canvass.

Election Verification : The combination of VoterVerification (cast-as-intended) and ResultsVerification (counted-as-cast) that provides full confidence that the Results accurately reflects the electorate's intent (counted-as-intended).

Election Verification Statement : See ElectionTranscript.

Electronic Voting Machine : See DRE.

Encrypted Ballot : The Ballots in the BallotBox.
See also VotedBallot and SignedVotedBallot.

Encryption Parameters : See ElectionParameters.

Hash : A unique mathematical fingerprint of some data.

Key Share : Pieces of a secret key used in a multi-party distributed key generation protocol, such as the PedersenSecretSharingProtocol.
See also TrusteePrivateKey.

LEO : Local election official.
See also ElectionOfficial.

LEO Key : LEO private key used to sign a BlankBallot and an ElectionTranscript.

L&A Test : See LogicAndAccuracyTest.

Logic and Accuracy Test : A battery of tests typically performed before and after an election to ensure that the election is, and was, configured properly.

Mark : See Answer.

Neff Shuffle Protocol : Neff shuffle protocol which enables VHTi ResultsVerification. See C.A. Neff, *Verifiable Mixing (Shuffling) of ElGamal Pairs*. http://www.votehere.net/vhti/documentation/egshuf.pdf.

See also Shuffle and ShuffleProof.

Observer : See ElectionObserver.

Pedersen Secret Sharing Protocol : Pedersen multi-party distributed key generation protocol. See T. Pedersen. *A threshold cryptosystem without a trusted party*, Advances in Cryptology - EUROCRYPT 91, Lecture Notes in Computer Science, pp. 522-526, Springer-Verlag, 1991.

PKI : *Public Key Infrastructure* for managing cryptographic keys with digital Certificates, typically conforming to the X.509 standard.

Poll Site : The place where the voter casts his/her Ballot.

Poll Worker : In the polling place, the person entrusted to authenticate voters, issue VotingTickets, and administer the DREs.

Poll Worker Computer : In the polling place, a computer used to generate VotingTickets, and possibly used for VoterRoll look-up and check-off.

Poll Worker Key : Key used by a PollWorker to digitally sign a VotingTicket after the voter has been appropriately authenticated, according to applicable law.

Polling Place : See PollSite.

Question : An issue or office to be voted upon.

Results : An assignment to each Answer, a non-negative total computed according to the TabulationRules. The *Results* may also contain additional data for the purpose of ElectionVerification.

See also ElectionResults.

Results Verification : Provides any ElectionObserver with the ability to verify with full confidence that all Ballots in the BallotBox were counted-as-cast. It involves three distinct checks:

1. Any ElectionObserver must be able to satisfy him or herself that every Ballot in the BallotBox comes from a unique, legitimate voter;
2. Any ElectionObserver must be able to satisfy him or herself that no voted ballots have changed since they were cast; and
3. Any ElectionObserver must be able to reproduce the election results.

Sealed Ballot Box : The BallotBox that is certified as final by appropriate authorities. It is not damaging for the *Sealed Ballot Box* to contain illegitimate or invalid Ballots (e.g., provisional ballots), since these will be detected and eliminated by the TabulationRules and Canvass. Since voters are able to detect and prove the condition of missing Ballots, policies should be in place that specify accountability.

See also SignedBallotBox.

Shuffle : A key component of the NeffShuffleProtocol which cryptographically mixes the EncryptedBallots before Tabulation, while still ensuring indisputable ResultsVerification, and providing another layer of privacy protection protection by separting the BallotSequenceNumber from the Ballot.

Once the polls close, the voted ballots are "shuffled" by the Trustees, in turn, to separate each uniquely identified ballot from the corresponding vote. This shuffle also produces verification data that proves that no ballots were added, deleted, or changed during the shuffling process. Once shuffled, all Trustees decrypt and tabulate the ballots to produce the plain text anonymous ballots for the ElectionOfficials to tabulate. This encourages the involvement of multiple Trustees representing a variety of allegiances.

See also Trustees and NeffShuffleProtocol.

Shuffle Proof : Auditable data produced by the NeffShuffleProtocol that proves that the underlying voter Choices did not change during shuffling of the EncryptedBallots.

See also Shuffle.

Tabulation : The counting of the Ballots to create the Results.

Tabulation Center : The site(s) where the Ballots are counted to produce the Results.
See also ElectionManagementComputer.

Tabulation Rules : The set of elementary arithmetic and logical operations that produce a unique Tally (see also Results) from any collection of Ballots. Requirements need not prescribe specific operations, but every system shall provide its own formal specification of *Tabulation Rules* satisfying:

1. The specification shall allow all ElectionObservers with access to both the collection of Ballots and the Tally data to determine, with full confidence, whether the Tally has been properly formed according to the *Tabulation Rules*.
2. The specification shall be publicly available.
3. *Tabulation Rules* shall satisfy an "additive property": if tallies for two disjoint collections of Ballots are each created according to the *Tabulation Rules*, the sum of the respective totals shall always be identical to the total created according to the *Tabulation Rules* for the BallotBox that is the aggregation.

See also ResultsVerification.

Tally : See Results.

Threshold : Minimum number of Trustees needed to perform security functions like ElectionConfiguration, Tabulation, and generation of VerificationStatements.

Touch-Screen Voting Machine : See DRE.

Trustee Private Key : Secret key held by a Trustee used for ElectionConfiguration, Tabulation and generation of VerificationStatements. See also KeyShare and PedersenSecretSharingProtocol.

Trustees : The election *Trustees* represent the public in safeguarding the election. They are typically representatives from groups of contrasting interest such as political parties, non-governmental organizations, and watchdog groups. The *Trustees* are cryptographically bound to act together for proper election operation, similar to when the launching of a missile requires the simultaneous turning of keys. A similar distributed trust structure is used to protect paper-based election systems.

The only way the *Trustees* might cheat is to engage in some form of vote selling or coercion. But even to do this, a Threshold (defined

before the election) number must conspire, which is unlikely given that they represent diverse allegiances. Note that they *cannot* conspire to compromise the election results without detection.

See also Trustee.

Verification Code : A small integer that corresponds to an Answer, used for VoterVerification. A different *VerificationCode* is assigned for each Answer for each BlankBallot.

Verification Codebook : A series of VerificationCodes for each BallotSequenceNumber. When the voter inserts their VotingToken into the DRE, the *Verification Codebook* is generated from the CodebookKey.

See also VoteVerificationCodebook.

Verification Receipt : Given to the voter in the voting booth, the *Verification Receipt* contains the Question and VerificationCodes that the voter has selected. It does not contain any Answers. The voter can then verify that the VerificationCodes on the *Verification Receipt* match the on-screen VerificationCodes. Once the voter is satisfied, the *Verification Receipt* is signed by the DRE using the DREPivateKey. The *Verification Receipt* may be fed through a reader or delivered in a format compatible for disabled voters.

After the election, the *Verification Receipt* is used to look-up the voter's VerificationCodes on the published VerificationStatementto ensure the ballot was cast-as-intended.
See also VerificationStatement, VoteReceipt, and VoteReceiptData.

Verification Statement : Published after the election, the *Verification Statement* allows the voter to verify that the VerificationCodes on his/her VerificationReceipt (provided to the voter in the voting booth) matches the VerificationCodes derived from his/her EncryptedBallot contained in the BallotBox, ensuring the ballot was cast-as-intended.

Depending on implementation, this look-up can be done by web-site, phone, or any other publishing mechanism using the BallotSequenceNumber on the VerificationReceipt.

See also VerificationReceipt and VoteVerificationStatement.

Vote Capture : The recording of VoterIntent on some storage medium.

Vote Casting : The placing of the Ballot into the BallotBox for Tabulation.

Vote Client : See DRE.

Vote Counting : The Tabulation of the Ballots in the BallotBox.

Vote Receipt : See VerificationReceipt.

Voted Ballot : A completed Ballot.

See also VotedBallot and SignedVotedBallot.

Voter Intent : How a voter wishes to vote.

Voter Registration Database : A paper or computerized master list of registered voters.

Voter Roll : List of registered voters used to authorize voters to vote. After the election, used in the official Canvass to verify the voters who received a VotingTicket.

Voter Verification : Provides the voter with the ability to verify, with full confidence, that his or her ballot was captured and cast as intended in the BallotBox. It involves two distinct checks:

1. The voter must be able to satisfy him or herself that the voted ballot is properly captured; and
2. The voter must be able to satisfy him or herself that the voted ballot is contained in the public BallotBox.

See also VerificationReceipt and VerificationStatement.

Voting Machine : See DRE.

Voting Ticket : A random number, digitally signed by a PollWorker with his/her PollWorkerKey, to authorize a voter to vote.

See also VoterRoll.

Voting Token : A token given to the voter to provide a VotingTicket. The token could be some piece of hardware (see DistributionMedium) or even a password.

Chapter 2

Data Definitions

The following are detailed data definitions used by the VHTi Election Verification technology.

Alphabet Encoding : 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).

Answer Mark : For the purpose of the cryptographic protocols, each answer (in the AnswerReference sense) must be "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.

Answer Partial Decrypt : A data structure of cryptographic quantities representing the piece of information contributed by a *single* Trustee

for a *single* VotedAnswer 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 VotedAnswer, (X, Y), with respect to an Trustee, A, with KeyShareCommitment, C, if and only if

$$c = H(g\,,\, C\,,\, X\,,\, Z\,,\, g^d C^c\,,\, X^d Z^c) \tag{2.1}$$

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

Answer Reference : 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 Text Structure : 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.).

Codebook Commitment :

An element of the ElectionEncodingSubgroup. Each CodebookCommitment 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 CodebookCommitments is (multi-dimensionally) indexed by all possible triples: VoteVerificationTrustee, BallotSequenceNumber, QuestionReference.

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

Ballot Answer : 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 Node : The data structure stored in the electronic ballot box as a result of a "successfully cast" ballot. It consists of the ValidatedVotedBallot received, and the PreVerificationCodes which was computed and returned. Of course, only the first element is critical, since the second can be computed from it, but storing the PreVerificationCodes is worthwhile to avoid potential load on the server from a (possibly malicious) voter who is determined to try to cast votes multiple times.

$$\text{BallotBoxNode} \doteq \left\{ \begin{array}{c} \text{ValidatedVotedBallot} \\ \text{VoteReceipt} \end{array} \right\}$$

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

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

Ballot Goo : Miscellaneous stuff – election name, titles, page and display info, etc., associated with the human readable elements of a ballot.

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

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

Ballot Question : 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 : A vector of L BallotQuestions

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

Ballot Secret : A secret value used to encrypt a ballot. It is an element of $\mathbf{Z}_q$.

Ballot Secrets : A vector of $\nu \geq 0$ secret values (elements of $\mathbf{Z}_q$), used to encrypt a ballot. This information is created by the Vote Client at encryption time, and should be carefully forgotten afterward.

$$\text{BallotSecret} \doteq (\alpha_1 , \ldots , \alpha_\nu)$$

Ballot Sequence Number : Ballot identifier used for poll site VoterVerification. 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 : A vector of V BallotSequenceNumbers

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

BSN Codebook Commitments : A structure that represents all CodebookCommitments for a *fixed* ElectionID, *fixed* PrecinctID, *fixed* VoteVerificationTrustee

and *fixed* BallotSequenceNumber. That is, a collection of $l \geq 1$ CodebookCommitments, where l is the number of QuestionReferences in the BlankBallot.

$$\text{BSNCodebookCommitments} \doteq$$

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

Blank Ballot : 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.

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

Broadcast Value : 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 : An XML string containing the BroadcastValue from each authority.

Certificate : A PKI, typically X.509, certificate.

Check Results : An XML structure containing the results of checking or verification.

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

Clear Text Ballot : 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 : An XML structure containing one or more ClearTextBallot.

Codebook Commitment :

An element of the ElectionEncodingSubgroup. Each CodebookCommitment 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 CodebookCommitments is (multi-dimensionally) indexed by all possible triples: VoteVerificationTrustee, BallotSequenceNumber, AnswerReference.

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

Codebook Verification Code : An element of the VoteVerificationCodebook which associates a particular VoteVerificationCode with the appropriate QuestionReference.

Committed Trustee :

$$\text{CommittedTrustee} \doteq \left\{ \begin{array}{c} \text{Trustee} \\ \text{KeyShareCommitment} \end{array} \right\}$$

Committed Trustee Set : A set of CommittedTrustee s

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

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

$$\text{CryptoGroupParameters} \doteq$$

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

Crypto Tabulation Parameters : 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.

$$\text{CryptoTabulationParameters} \doteq \left\{ \begin{array}{c} \text{ElectionPublicKey } (h) \\ \text{SecEncryptionBase } (h_0) \\ \text{CommittedTrusteeSet } (\textit{All } n \textit{ Tabulation Authorities}) \\ \text{TabulationThreshold } (t) \end{array} \right\}$$

Decryption Validity Proof : A zeroknowledge proof of correctness for decryption.

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 : 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 : The modular arithmetic group specified by the ElectionModulus (p) parameter of the CryptoGroupParameters structure contained in the CryptoElectionParameters structure of the BlankBallot.

Election ID : A UUID for elections.

Election Modulus : 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 Node : The data structure encompassing the (unsigned) data loaded by LoadElection. (It also needs to have pointer information to allow for efficient insertion of ValidatedVotedBallot.)

$$\text{ElectionNode} \doteq \left\{ \begin{array}{c} \text{ElectionID} \\ \text{CryptoElectionParameters} \end{array} \right\}$$

Election Public Key : An element, h, of the ElectionEncodingSubgroup. The corresponding *private key* ($\log_g h$, where g is the ElectionSubgroupGenerator) is a secret cryptographicly shared between a set of "election trustees" (TrusteeSet) 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 Generator : A fixed element, g, of the ElectionGroup which generates the ElectionEncodingSubgroup. In particular, g must satisfy the following relationship with the ElectionSubgroupModulus, q:

$$|g| = q \tag{2.2}$$

Election Subgroup Modulus : 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:

$$\begin{aligned} p - 1 &= q\,r \\ (q\,,\,r) &= 1 \end{aligned} \tag{2.3}$$

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

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

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

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

Encrypted Session Key : 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 : A key which can be used for decryption.

Encryption Public Key : A key which can be used for encryption.

Error Structure : Structure for encoding "unexpected" return conditions.

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

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

Identification Information : An XML string containing identifying information about the owner/creator of a GeneralPurposePublicKeyor GeneralPurposePrivateKey.

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

Key Generation Parameters : The parameters determining the number of Trustees (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 : 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 Trustees officially participating in Key Sharing.

Keys : A vector of *key* items.

Modular Integer : A BigInt

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

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

Multi-Set Element : A data pair

$$\left\{ \begin{array}{l} \text{ModularInt } \gamma \quad \text{(an element of ElectionEncodingSubgroup)} \\ \texttt{int} \text{ count} \end{array} \right\}$$

Pair-wise Secret : Each Authority evaluates his polynomial at all of the TrusteeEvaluationPoint 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 Trustee object.

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

Partially Decrypted Ballot Box : 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 is needed to certify that the count is derived properly from the official set of SignedVotedBallots.

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

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 Codebook Commitments : A structure that represents all CodebookCommitments for a *fixed* ElectionID, *fixed* PrecinctID. That is, a collection of $N_{VVT} \geq 1$ TrusteeCodebookCommitmentss, where N_{VVT} is the number of VoteVerificationTrustees for the Precinct indicated by (ElectionID, PrecinctID) .

$$\text{PrecinctCodebookCommitments} \doteq (\text{TrusteeCodebookCommitments}_1, \ldots, \text{TrusteeCodebookCommitments}_{N_{VVT}})$$

Precinct ID : 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 : 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 : A vector of $\nu \geq 0$ PreVerificationCodes:

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

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

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

Question Reference : 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 Text Structure : 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 : A collection of random, or pseudorandom bits.

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

Random IJ State : 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 : A short byte sequence used to seed the random-number generator.

Random State : 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

$$\text{RandomState} \doteq (\textsc{nil})$$

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

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

Raw Question Results : A pair

$$\text{RawQuestionResults} \doteq \left\{ \begin{array}{c} \textsc{question id} \\ (\text{MultiSetElement}_1 , \ldots , \text{MultiSetElement}_Q) \end{array} \right\}$$

where the second element is a vector of multi-set elements – one for each allowed answer to the question.

Raw Results : A vector of question results

$$\text{RawResults} \doteq (\text{RawQuestionResults}_1 , \ldots , \text{RawQuestionResults}_B)$$

Raw Voted Ballot : A vector of $m \geq 1$ *Voted Answers*:

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

Result Verification Trustee : Currently a synonym for Trustee.

Secret Coefficients : Cryptographic quantities specific to the Key Sharing protocol.

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

where

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

Secondary Encryption Base : An additional element of the ElectionEncodingSubgroup used for ElectionVerification.

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

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

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

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

Shuffle Validity Proof : A zeroknowledge proof of correctness for shuffle.

Signature : A data string which may be used to ensure that another string was created, or endorsed, by a person whose GeneralPurposePublicKey we have.

Signed Ballot Box : 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 : 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 : An XML structure containing a hash of the original plaintext to be signed, and a Signature.

Signed Election Parameters : 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 Status Query Structure : A signed StatusQueryStruct.

Signed Status Response Structure : The format for "secure" replies from the Vote Kernel.

Signed Voted Ballot : A *Voted Ballot* with detached (voter) signature.

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

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

Signing Private Key : A key which can be used for signature generation.

Signing Public Key : A key which can be used for signature verification.

Status Query Structure : A (XML) structure for encoding a state, or status query passed to the Vote Kernal. This structure will likely include

- a "status type" enum to specify the type of query
- a challenge RandomBits (to prevent replays)
- an ElectionID (which may be NULL)
- a VoterID (which may be NULL)

Tabulation Threshold : A positive integer (t) which specifies the number of CommittedTrustees 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 CommittedTrusteeSet structure of the CryptoTabulationParameters structure contained in the CryptoElectionParameters structure of the BlankBallot.

Trustee : The data structure identifying a *Trustee* who has, in advance of the election, been appointed to "oversee" the election is

$$\text{Trustee} \doteq \left\{ \begin{array}{c} \text{Certificate} \\ \text{TrusteeEvaluationPoint} \end{array} \right\}$$

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

$$\text{TrusteeCodebookCommitments} \doteq$$

$$\left\{ \begin{array}{c} \text{Trustee} \\ \text{BlankBallot} \\ (\text{BSNCodebookCommitments}_1, \ldots, \text{BSNCodebookCommitments}_{N_{BSN}}) \\ \text{Signature} \end{array} \right\}$$

Trustee Evaluation Point : A modular integer β where

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

Trustee Partial Decrypt : The data structure representing the decryption information contributed by a *single* CommittedTrustee.

$$\text{TrusteePartialDecrypt} \doteq \left\{ \begin{array}{c} \text{CommittedTrustee} \\ \text{BallotBoxPartialDecrypt} \end{array} \right\}$$

Trustee Partial Decrypts : An XML structure representing a *set* of one or more PartiallyDecryptedBallotBox. The order of the elements of this set is irrelevant.

$$\text{TrusteePartialDecrypts} \doteq \{\text{TrusteePartialDecrypt}_1 , \ldots , \text{TrusteePartialDecrypt}_t\}$$

Trustee Set : A set of Authorities.

$$\text{TrusteeSet} \doteq \{\text{Trustee}_1 , \ldots , \text{Trustee}_t\}$$

UID : Unique identification number.

UUID : Universally unique identification number.

Validated Voted Ballot : A *Signed* Voted Ballot (SignedVotedBallot) along with a vector of *Answer Validity Proofs* (AnswerValidityProof). The vector of AnswerValidityProofs should correspond directly with the RawVotedBallot in the SignedVotedBallot. That is, the RawVotedBallot is a vector of Voted Answers (VotedAnswer), and the i^{th} AnswerValidityProof should be a proper validity proof for RawBallotBox[i].

$$\text{ValidatedVotedBallot} \doteq \left\{ \begin{array}{c} \text{SignedVotedBallot} \\ \text{AnswerValidityProof}[m] \end{array} \right\}$$

(Note that the AnswerValidityProof array could be NULL on ballot submission if the voter wishes to opt out of VoterVerification.)

Vote Receipt : 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 : 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.

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

Vote Signing Certificate : A Certificate corresponding to one of the VoteSigningKeys used in the election. In the case of remote voting, this Certificate is exactly a (registered) voter Certificate. In the case of poll site voting, all VoteSigningCertificates *must be published* for the purpose of election verification prior to the start of vote casting.

Vote Signing Certificates : A vector of VoteSigningCertificatess.

$$\text{VoteSigningCertificates} \doteq (\text{VoteSigningCertificate}_1 , \ldots , \text{VoteSigningCertificate}_n)$$

Vote Signing Key : A SecretKey used for signing a voted ballot. In the case of remote voting, this is exactly the private key corresponding to a voter's Certificate. In the case of poll site voting, each voting machine must have a VoteSigningKey for the purpose of ballot encryption. Whether or not the same key is used for multiple machines is left as a policy decision.

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

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

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

Vote Verification Codebook : Data that assigns a VoteVerificationCode to each AnswerReference on the BlankBallotfor a fixed VoterID or BallotSequenceNumber. It is required that 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. It should be noted that usually VoteVerificationCodebooks are computed from a collection of VoteVerificationCodebooks (or VoteVerificationKeys that represent them).

Vote Verification Codebook Share : Data generated by an individual VoteVerificationTrustee for the purpose of creating a VoteVerificationCodebook with shared trust characteristics. For the purpose of minimizing the amount of secret data that must be stored by each VoteVerificationTrustee, it is possible to associate a single VoteVerificationKey with a full set of VoteVerificationCodebookShares via a fixed pseudo-random process.

Vote Verification Key : A SecretKey used by an individual VoteVerificationTrustee to pseudo-randomly generate CodebookCommitments.

Vote Verification Keys : A vector of VoteVerificationKeys.

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

Vote Verification Statements : A collection of VoteVerificationStatements.

Vote Verification Trustee : Currently a synonym for Trustee.

Voted Answer : An ElGamal pair encrypting the voter's chosen *Answer Mark*.

$$\text{VotedAnswer} \doteq \left\{ \text{ ElGamalPair } \right\}$$

Voted Ballot :

$$\text{VotedBallot} \doteq \left\{ \begin{array}{c} \text{ElectionID} \\ \text{VoterID} \mid \text{BallotSequenceNumber} \\ \textsc{hash}(\text{BlankBallot}) \\ \text{RawVotedBallot} \end{array} \right\}$$

Voter ID : UUID for voters.

Voter Roll : This data structure is needed for ballot authentication and deduplication. Essentially it is a vector of certificates (Certificate) corresponding to the set of eligible voters.

$$\text{VoterRoll} \doteq \left\{ \begin{array}{c} \text{Jurisdiction ID Info} \\ (\text{Certificate}_1 , \ldots , \text{Certificate}_N) \end{array} \right\}$$

