votingapp

End-to-End Verifiability

with the VotingApp

Version 1.0 - 8/6/2021

Authors: David Wallick, Ryan Cook and Greg Long

1 Abstract

Online voting for public elections in the United States is not a widely accepted technology although many attempts at developing systems or protocols for doing so have been attempted over the years. With the latest set of federal standards explicitly disallowing voting over public networks the potential for getting such a system certified for use with the US is reduced significantly. As a result, initiatives to create such systems may also be slowed. However, using some well defined existing standards and definitions of end-to-end-verifiable voting systems one can still attempt to innovate on the pieces necessary to make voting over public networks possible.

Vidaloop's VotingApp attempts to solve the problem of data verifiability while maintaining voter privacy. This is achieved through a combination of a custom hardware device and specific software operations to create the cryptographic root of trust for the system and a series of intermediate certificates used to verify data as it is passed between components of the system including the voter device. This root of trust branches out to all other components such that all data transfers can be verified to be authentic to the system and unaltered at any point. Voter anonymity is retained by keeping voter records only within an air-gapped security device, inaccessible to all users except when used to run specific elections operations. Combined with a Hierarchical Deterministic Address Protocol (HDAP) for storing Cast Vote Records (CVRs), voters, administrators, and observers have multiple methods for data integrity and accuracy verification.

The core protocol proposed here along with some details around necessary hardware components for administering the election with Vidaloop VotingApp could potentially open doors to online voting solutions previously closed in the US public elections market. The paper focuses on the core protocol and the necessary hardware components only and leaves open discussion around implementation details.

Definitions and Terms

Definitions of terms used in the paper can be found in Appendix A

2 Introduction

Public elections in the United States consist primarily of poll station in-person voting either on paper ballots or on highly controlled and monitored electronic machines. Vote-by-mail paper ballot voting has not been widely adopted except in a few states prior to the 2020 election. Overseas voters covered by the Uniformed and Overseas Citizens Absentee Voting Act (UOCAVA) vote primarily on paper even if portions of the process are done electronically.[1,2] While these methods have been refined for security, accessibility, and privacy concerns, they still have their drawbacks.

For example, paper ballots offer high verifiability and accuracy, especially in post election audits, but are difficult to use for certain voter populations. Frequently security of paper systems relies on physical security, or on process procedures that work well but focus primarily on tamper prevention principles and cannot provide direct ballot-by-ballot verifiability to individual voters. None of these systems offer voter verifiability from beginning to end of the voting process.[3]

In the late 90's many ideas have circulated on using the internet for voting, with goals of providing increased convenience and access to voters, as well as potentially reducing administrative costs to jurisdictions.[4,5,6] Ross Perot's Reform Party might have been the first US political party to employ online voting, in 1996.[7] However, large scale voting over public networks in government sponsored elections has not been a practical possibility in the United States due to security concerns, privacy concerns, and a general lack of verifiability. Voters, election officials and outside observers have not been able to verify the integrity of online elections.[8,9]

Experimentation in remote electronic voting occurred in the early 2000s but those efforts rarely led to use in public elections and never at wide scale. Recent years have also seen a boom of election vendors experimenting with public network voting, but the sentiment of such efforts has been met with justifiable skepticism after numerous systems and implementations contain serious or critical flaws in technology and/or implementation details.[10,11,12,13] The latest federal voting system standards document, Voluntary Voting System Guidelines 2.0 (VVSG 2.0), has explicitly excluded voting via public networks as a

result of the concerns but does make reference to designs 'to meet the challenges ahead'.

The Guidelines were designed to meet the challenges ahead, to replace decades-old voting machines, to improve the voter experience, and provide necessary safeguards to protect the integrity of the voting process. All sections of the prior VVSG versions have been reviewed, reevaluated, and updated to meet modern expectations, which address how voters should interact with the voting system and how voting systems should be designed and developed. The VVSG 2.0 requirements represent the latest in both industry and technology best practices, requiring significant updates in many aspects of voting systems. The Guidelines allow for an improved and consistent voter experience, enabling all voters to vote privately and independently, ensuring votes are marked, verified and cast as intended, and that the final count represents the true will of the voters.[14]

Additionally the VVSG has an entire section dedicated to cryptographic end-to-end verifiable voting systems, defined by the VVSG as the following:

A voting system that uses cryptographic techniques to store an encrypted copy of the voter's ballot selections while maintaining ballot secrecy and allows election outcomes to be independently and universally verified by members of the public. These voting systems provide voters with a special receipt of their cast ballot—one that allows them to verify their vote was included in the outcome but does not reveal to anyone how they voted.[15]

While the reference within the VVSG is specific to End-to-End Verifiability (E2EV) within poll-station voting, the concepts are valuable to apply to networked voting systems as well. Using that document as a starting point for defining E2EV, this white paper presents the Vidaloop solution for providing end-to-end verifiability in a public network voting system.

2.1 Problem Statement

The challenge of voting on public networks is combining the necessary security and vote verifiability for voters with the need to maintain individual voter privacy with regard to specific selections/votes.[16] Many non-voting-related electronic systems can use public networks because privacy of the end user is permitted to be tied directly to the data being transferred and secured on either end. Additionally, as in banking online, corrections after the fact can be made to transactions should errors or interference be discovered at some later point. These principles do not apply to online voting and therefore, systems must attempt to provide data security and verifiability in real-time to voters, election officials, and outside observers, but without revealing the actual vote contents of any given voter until such a point that voter cannot be definitively tied to the selections.

Because of the privacy requirements, keeping data verifiable and secure throughout the process is a constant challenge for all voting systems but particularly difficult for online voting systems. This is true because paper systems can rely heavily on process and physical security, especially during the vote submission and storage activities, that remote electronic systems cannot leverage. Any online voting system needs to maintain a higher level of voter verifiability, especially after vote submission, than any paper-based system relying on process and physical security of ballots. This added need for verifiability all the way through vote aggregation and tabulation creates the greatest conflict with privacy requirements.[17]

2.2 Background

Voting systems dependent solely on paper are particularly difficult to verify end-to-end, especially in remote voting. As discussed in their handout on verifiable voting, Benaloh, Rivest, et al recognize this challenge with regard to ballot transport and storage.

Verifiably secure transport and storage of paper records is particularly challenging for remote voting, whether in a supervised or unsupervised polling place.[14]

The solution to the problem is to attempt to supplement or in some cases replace the paper process with a digital process or digital data that can be verified cryptographically in all stages of an election. Several designs have been proposed that follow the basic cryptographic E2EV principals but each solves the same problems in different ways with varying degrees of success..

Before we review some other protocols/systems, let's define end-to-end verifiability. From Benaloh, Rivest, et al the following description is used for the most basic definition of E2EV systems:

End-to-end verifiability is a collection of techniques for replicating, and in some ways exceeding, the standards of evidence provided by an ideal, observed, polling place…

1. ***Cast As Intended**: voters make their selections and, at the time of vote casting, can get convincing evidence that their encrypted votes accurately reflect their choices;*
2. ***Recorded As Cast**: voters or their designates can check that their encrypted votes have been correctly included, by finding exactly the encrypted value they cast on a public list of encrypted cast votes; and*
3. ***Tallied As Recorded**: any member of the public can check that all the published encrypted votes are correctly included in the tally, without knowing how any individual voted.*[14]

2.2.1 Review of Existing E2E Systems/Protocols

A number of E2EV systems have been proposed, including some which have been implemented in limited deployments. The following is a list of a few of the more successful or well studied protocols/systems. This list is not exhaustive.

ThreeBallot

An E2EV auditable voting system that can be implemented on paper. It attempts to be both anonymous and verifiable by giving each voter three ballots, two which are anonymous and a third which is verifiable. The voter chooses which ballot is verifiable and keeps this secret; since the vote-counter does not know, there is a 1/3 chance of being

discovered modifying any single ballot. The voter is forced to make two of their three ballots cancel each other out, so that they can only vote once.

An electronic version of this model addresses the complexity and usability issues necessary to ensure the scheme is unbreakable. [18]

Pros: Auditable and anonymous

Cons: Extremely complex vote marking process, requires the voter to correctly cancel out two of the three ballots. This canceling process results in voter ability to manipulate the system for a double vote possibility that may go undetected at least until tally has occurred and is too late to correct if at all. Additionally it cannot support Ranked Choice Voting.

Punchscan

Uses paper ballots, where votes are recorded using two layers of paper. The top layer includes the candidate names or measures along with a letter or symbol next to the name. The order of the letters is randomly generated for each ballot. Below the candidate list are a series of holes. The second layer contains corresponding printed letters, also randomly generated. When selecting a candidate, the voter marks both the exterior of the hole on the top layer and the matching spot on the lower layer. The voter then separates the ballot, choosing either the top or bottom layer as the receipt, which is then scanned for tabulation. The other layer is shredded. The voter cannot prove to someone else how they voted, which prevents vote buying. [19]

Pros: Removes the possibility of the voter proving how they voted.

Cons: Requires paper ballots. Accurately marking the ballot could be difficult.

Prêt à Voter

Each ballot contains a randomized list of candidates, along with a unique code printed at the bottom. After voting, the voter separates the ballot, removing and discarding the list of

candidates from the marked votes. The marked list with the code is scanned for tallying. Information to decrypt the code is spread among several different tellers, requiring the tellers to act together to interpret the vote. [20]

Pros: Vote decryption requires a quorum of tellers, preventing a single bad actor from affecting the total

Cons: Requires a paper ballot; hard to implement remotely if at all

Scantegrity and Scantegrity II

These provide a security enhancement for optical scan voting systems using confirmation codes to allow the voter to verify that their ballot is included as voted in the final tally. Scantegrity II includes printing the confirmation code in invisible ink, requiring the voter to use a special pen which causes the confirmation code to appear. After the election is finished, the election authority publicly posts a list of confirmation codes for the positions marked on each ballot it received. Voters who wrote down their codes can verify that the codes are correct for their ballot number and that no codes were added or removed. [21]

Pros: Voter can verify their votes were counted

Cons: Vote verification can be used to buy or coerce votes. Requires paper ballot.

ElectionGuard

ElectionGuard provides E2E verifiability, allowing individual voters to verify that their votes have been recorded correctly, and allowing observers to verify that the votes have been counted accurately. ElectionGuard can infer that the voter entered a write-in candidate but cannot find what they wrote in. It requires a paper ballot to be collected, preventing its use as a completely remote voting solution.

During an election or audit, ElectionGuard will encrypt the completed ballot and return a verification code to the voter. The encrypted ballots are published along with

non-interactive zero-knowledge proof of their integrity. The encryption method has a homomorphic property which allows the encrypted ballots to be combined into a single aggregate ballot used for tally purposes. Individual ballots are never decrypted.[22]

Pros: Voter privacy is maintained in full and voters can prove their vote was not changed. Multiple guardians are required to verify the tally, preventing a single bad actor from affecting the total.

Cons: A paper ballot is required; Ranked choice voting is not supported; Write-ins must be explicitly registered making non-certified write-in support difficult

These systems were chosen from a list of many as examples of systems/protocols that have solved many of the same problems. As the review shows many of the systems significantly sacrifice usability especially around the vote marking process to achieve voter verifiability. However, many of the ideas from these other systems, especially around the cryptography and verification methods used serve as useful building blocks for understanding the Vidaloop solution.

3 Product Technology Overview

Vidaloop VotingApp uses a variety of well-established hardware and software cryptographic technologies. Included is a commercially available cryptographic hardware chip designed specifically for securing keys and sensitive data, signing and encrypting data with industry standard ciphers, standard certificate chain of trust processes, and the use of blockchain as the ballot box public ledger. Multiple redundant and complementary security measures are utilized to mitigate potential unknown security vulnerabilities.

See Appendix D for a list of technologies used.

4 Solution

Vidaloop VotingApp is an E2EV mobile voting system that does not require poll station voting or paper ballots. Using a combination of security certificates, blockchain, and other cryptographic technologies, it creates a robust E2EV solution that is secure over public networks, allowing users to vote remotely using their own smartphones or tablets. By controlling the data, it is not necessary to control the hardware used by the voters.

The typical E2EV definition described previously, involves three key verifications.

1. ***Cast as Intended Proof -*** *Voters can verify their vote is cast as they marked their ballot.*
2. ***Recorded as Cast Proof -*** *The voter and observers can verify the authenticity and integrity of each vote.*
3. ***Tallied as Recorded -*** *All observers of the system can verify all valid votes are included in the final count*.

Later in the paper we will explore each of these verifications as they relate to individual and aggregated votes. First, a description of the network construction and data transfers is required to better understand how the system itself can trust all pieces of data being passed between components.

As mentioned, the concept of internet voting has existed nearly as long as the internet itself, and many attempts have been made to create public-network-based systems. As of yet, they have all failed either in security implementation details, maintaining voter privacy, or usability. Systems that have solved many of the security concerns while maintaining privacy often ask a lot from the voter. As an example, the use of code voting techniques to allow for voter verification of choices without revealing actual selections often puts an undue burden either in registration and pre-election process on the voter (as in the VeryVote solution[23]) or require a voter to 'remember' potentially obscure bits of information for verification purposes later. For this reason Vidaloop felt it was necessary to focus on a different approach than simply solving the E2EV proofs previously mentioned.

By focusing first on building the trust between components and thus data transfers between them and then on the three E2EV proofs, Vidaloop VotingApp provides a more complete practical application of online voting than many of its predecessors.

4.1 Component Descriptions

A short description of the components involved in Vidaloop VotingApp is provided to help understand the diagrams and data flows presented after.

4.1.1 Security Device

The Security Device is a critical component in creating a secure election while still being able to maintain privacy of the voters. It maintains the root of trust for the election as well as being the only place in the system that keeps a tie between identifiable voter information and the data used to secure all voter transactions within the system. The Security Device is a purpose-built air-gapped hardware device owned and maintained by the administering jurisdiction. It is provisioned by Vidaloop at the time of assembly such that it can be verified to be a Vidaloop trusted device when delivered to customers. See *Section 4.4 Provisioning Components* for more details.

The Security Device uses a special purpose security chip to secure its private keys and to perform cryptographic operations. The private keys and certificates are created and locked during the provisioning phase of manufacturing. Once locked, the data in the chip cannot be modified and the private keys cannot be read.

Physical security measures such as limiting port access with the use of internal switches as well as tamper evident measures on the encasing of the device ensure any attempts to intrude into the Security Device would be either prevented or detected. Of all components within the system, the Security Device must be protected with the highest levels of physical, process, and software security measures.

The device has four main roles that it plays in the election system.

- **Root of Trust**
  As the root of trust for the election system, the device securely stores the election body's signing and encryption certificate, acts as an intermediate certificate authority, and signs data packages and records so they can be verified by other participants.

- **Secure Data Storage**
  The device stores data securely yet still allows an election administrator to perform critical operations on the data without revealing any secrets or compromising voter privacy.

- **Secure Data Processing**
  The device provides secure data processing, including:
  - Election data signing
  - Voter cryptographic key generation
  - Creation and signing of authentication keys of system components
  - Post election verification
  - Adjudication operations

- **Mediation of Human Processes**
  In order to help election administrators run a secure and safe election, the device will mediate, verify, and audit the human elements of the process. It accomplished this in a few different ways:

  1. **Required Physical Access**
     The device does not contain any way to be connected to a computer network and as such will require users to be physically present and authenticated with the device before performing any operations.

  2. **Quorum Validation**
     Any of the secure data processing operations can be configured to require a quorum to complete. When configured ***n*** number of people will have to be present and physically authenticate with the device for specified operations.

3. **Human decisions without exposing secret data**
The device allows election administrators to enforce rules and make decisions in a way that will not compromise the privacy of the voters. An example would be the reconciliation of ballots collected in Vidaloop VotingApp against ballots collected through other methods such as mail-in votes or in-person poll station voting. By adding a list of voters who voted using other methods to the Security Device after the polls close, the device can produce a list of VotingApp ballots to be disregarded in the ledger, in instances where someone has attempted to vote both in the VotingApp and with other methods. This calculation does not reveal the identity of the voter when removing the ballots.

4.1.2 Election Management System

The Election Management System (EMS) is used to create and maintain election definitions, election infrastructure settings, the voter list, and ballot data. It is responsible for verifying the health of the election network once deployed. It connects to every piece of the system (except voter devices) via networks and/or manual data transfers. Due to its centralized role in the network the EMS is provisioned after the Security Device but before all other components and can only be paired to a single set of Security Devices (a main Security Device and its associated backup devices). Similarly once provisioned to an EMS, any given Security Device cannot be provisioned with another EMS. This one-to-one relationship exists because both devices are expected to be used for multiple elections operated from a single physical location by a very small set of approved jurisdiction users.

4.1.3 Election Registry/Language Pack Service

The Registry server contains election ID information and other election-specific information including supported languages, and is the access point for the ballot box ledger. Before the mobile application knows which Public Ledger endpoint to access, it requests and receives necessary information

from the Registry. The Registry does not initiate contact with other components, but rather holds data and delivers when requested.

4.1.4 Public Ledger

Vidaloop VotingApp leverages blockchain technology to store the election voter list for the purpose of voter authentication and ballot data used to deliver the proper ballot style to each authenticated voter. This same blockchain ledger collects votes from the voter, serving as the digital ballot box. The ballot styles, a form of the voter list with no voter personally identifiable information, and election settings are stored on the blockchain. The blockchain records a limited number of transaction types to keep the processing needs of the blockchain limited, thus increasing performance capabilities.

The key feature of the component is its immutable nature, which makes it ideal for operating as the election progress ledger. The blockchain component is election specific and as such will be deployed and provisioned with new keys and certs for each election cycle. While the blockchain can remain up indefinitely post election for audit and posterity purposes, it is only active for data collection during designated times set by the election administrator from that election's associated EMS.

Administration of blockchain nodes can be distributed between multiple Trustees and deployed against multiple platforms reducing the opportunities for collusion or consensus attacks that permissioned blockchains can be vulnerable to when used in election scenarios.

4.1.5 Voting Application

The mobile voting application runs on voter owned and controlled mobile devices. It is used to initiate the voter authentication process, receive the properly assigned ballot, mark the ballot by the voter, and return the encrypted marked ballot and associated cryptographic receipt. The application does not

have system security keys because election officials cannot provision voter owned devices each election. However, installed in the application during the software build time is the Vidaloop Root Certificate that is common to all other components. When the app is downloaded from the application store for the user's device (e.g. Google Play Store), the certificate is downloaded as well. In this manner even the voter owned device becomes a component within the network loop of data transfers.

4.1.6 Voter Token Management Service

The Voter Token is an essential piece to creating the authentication of the voter within the system without compromising their identity when tied to a returned ballot. This token contains a few pieces of critical information for the system. First is the location of the Registry and Blockchain components such that the voter's device running the mobile voting application can identify where to send initial requests both for getting election settings and for verifying connection to the proper election network. Secondly, the token contains a random bit of data that, when used with pre-determined voter credentials, can create a unique cryptographic asymmetric key pair on the voter device for the purpose of signing all requests sent to the blockchain. Alluded to previously is a voter list on the blockchain (in the form of public keys generated from the same data and an associated ballot style ID, created by the same key generation library within the Security Device). When a voter creates their private/public key pair (PPK) and sends an authentication request to the blockchain signed by the private key, the blockchain can verify the request with the public key, thus closing an authentication loop without any voter details being stored anywhere but internally on the air-gapped Security Device. Because this process is extremely sensitive with regard to maintaining voter privacy, the delivery of this token must also be secured.

4.2 Creating a Certificate Chain

A core element of the VotingApp system is the Security Device. It is provisioned at manufacturing time with a Signed Root Certificate from the securely stored Vidaloop private key.

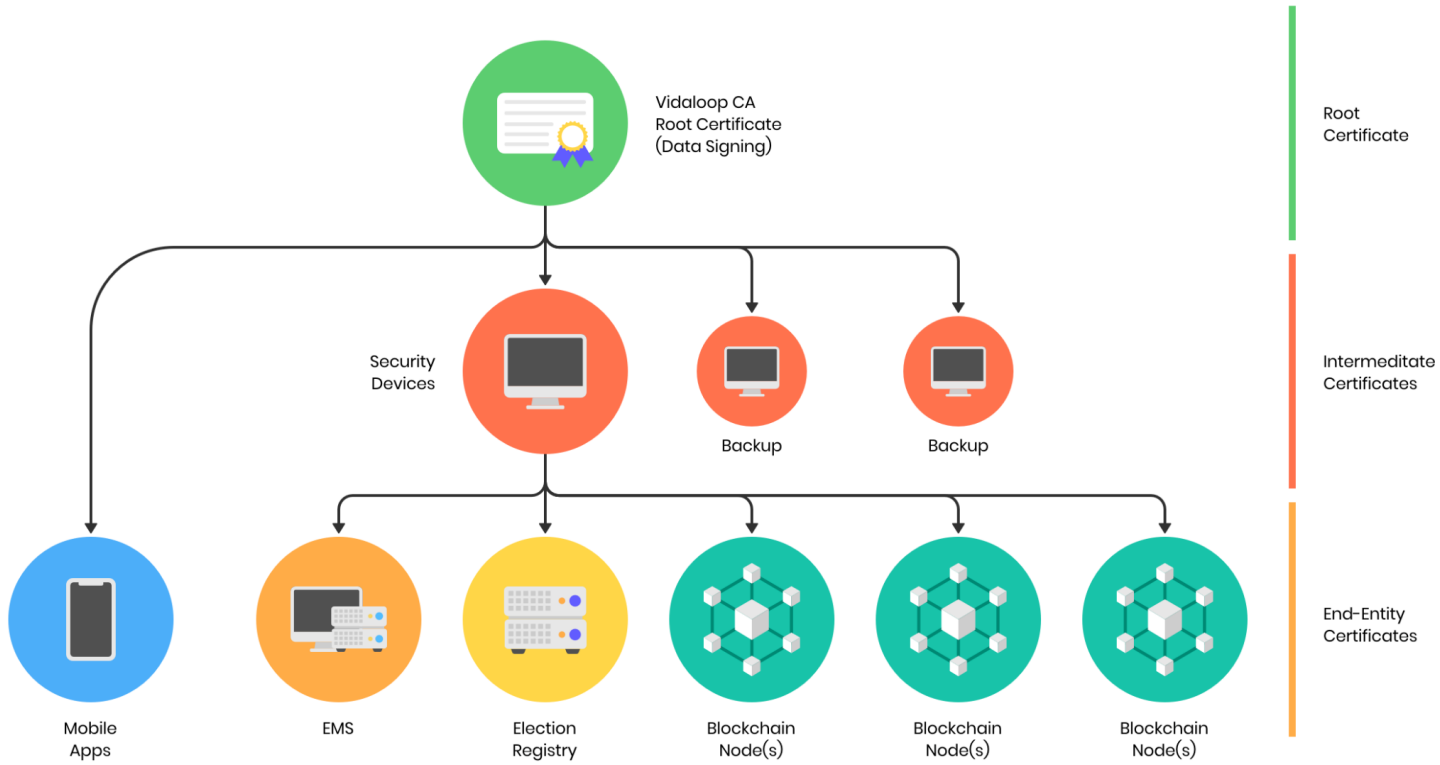

Figure 1: Certificate Distribution

Delegation of Trust

Trust is delegated to the Security Device which in turn delegates trust to various election components by acting as a certificate authority. In most cases, trust is established by verifying that the source of the data has a certificate signed by the same Security Device. The exception is the mobile Voting Client. The mobile application will confirm that it can trust the system or data by examining the chain of trust and confirming that the root of trust is a trusted source (Vidaloop).

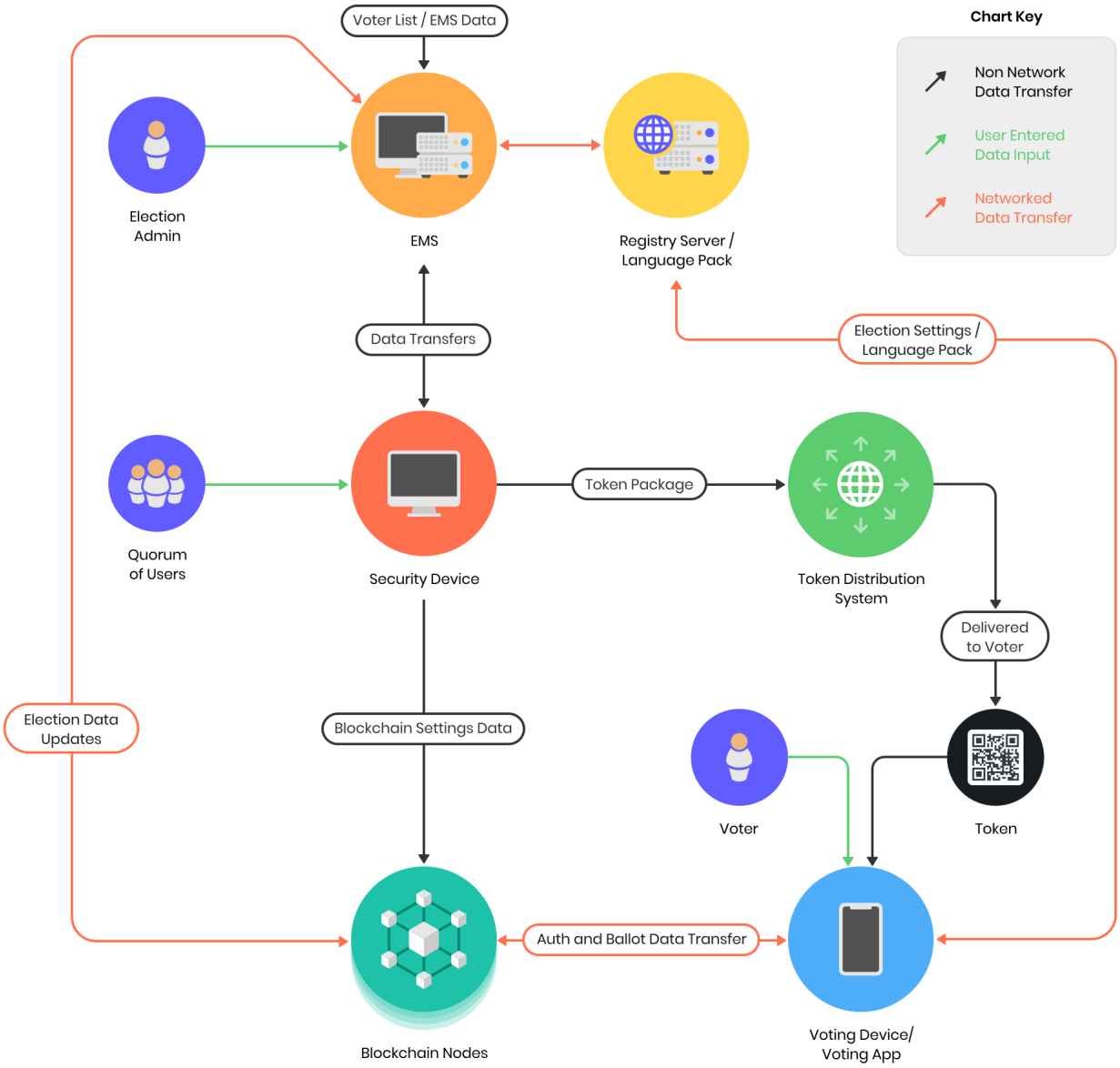

Figure 2: High Level System Data Flow

4.3 Data Flow High Level Description

Figure 2, above, depicts a fully provisioned and operating election network. The communication paths shown represent data transfers during pre-election, live election, and post-election phases. It depicts the connections between components and distinguishes between networked traffic and manual data transfers. For a detailed view of data flows see ***Section 4.6***

4.4 Provisioning Components

To create the network depicted above the components must be provisioned in a specific order. Provisioning this way allows each component to share certificates with one another in order to verify data transferred between one another. Some provisioning is done once in the lifetime of the component while other provisioning is done on an election by election basis. This will be called out for each component. See ***Appendix C*** for the specifics of component provisioning.

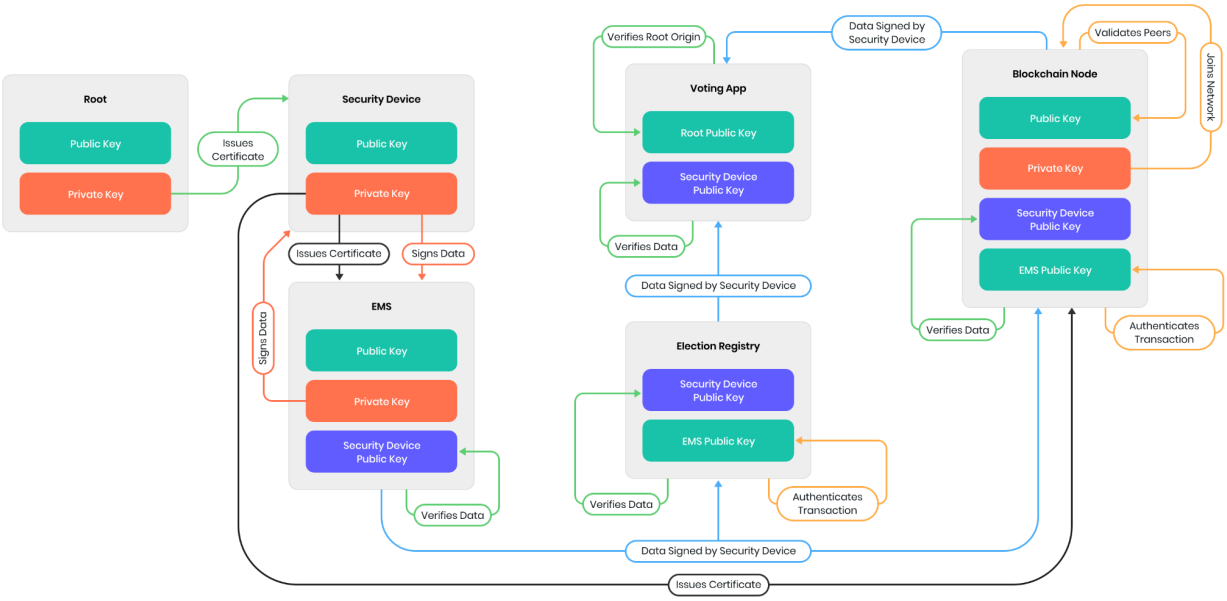

Figure 3: Data Verification

4.5 Establishing Data Verification

Data within the system is stored and transported in such a way that the origin of the data can be traced and the contents cannot be tampered with. Data is verified by confirming that it is signed by a trusted source. Using the various keys and certificates provisioned in the patterns shown in the diagram below, the end-to-end data verification can be established. Trust is verified by following back the entire chain to the root certificate located in the Security Device. See *Figure 3*, above.

4.6 Detailed Data Flows

Part of the security and verification of Vidaloop VotingApp is that no data can go between components without being checked for authenticity and integrity. While this doesn't completely prevent nefarious activity due to the fact that many components have ownership/maintenance responsibilities spread among a variety of possible trustees it does mean that all components would need to be compromised together in order for disruptions to data authenticity or integrity to occur. This should be caught in real time if any single component begins misbehaving.

4.6.1 Election Network Detailed Data Flow Diagram

The now provisioned election network passes different data from various components through both networked and manual methods. The following diagram and numbered description indicates the types of data being transferred and method of transfer. Each time a piece of data is prepared for export from one component it is signed with that component's private key. Upon retrieval of any data package every component will verify signatures with the public certs provided during provisioning. In some cases when a component acts as a pass through without altering data (example, the EMS receiving an output from the Security Device for delivery to the Registry) it will still sign the package thus creating two signatures the end receiving component must verify. These data flows described here are all inclusive across all stages of the election (pre-election; live voting; and post-election activities). *Sections 4.6.2 and 4.6.3* provide greater detail on some generalizations shown here.

Workflow Steps (See *Figure 4*)

1. Election data imports into EMS (as needed)
2. Election official edits/adds data inputs (as needed)
3. Election Export signed by EMS private key sent to Security Device via secure USB drive
4. Security Device quorum of users initiates create election process
5. Security Device exports package for delivery to EMS via secure USB drive of Registry data signed by the Security Device private key and a package of Blockchain seed data signed by the Security Device private key
6. Security Device exports package for delivery to Token Distribution vendor of Token Delivery Package signed by Security Device private key

7. Tokens created in final form and delivered to voters
8. EMS signs Registry data and delivers via network connection to Registry; health checks can be returned
9. EMS signs Blockchain seed data and delivers to Blockchain network via established network connection; health checks returned as well as post-election data sets upon EMS request.
10. Blockchain data propagates across all nodes
11. Token scan and voter inputs
12. Voter requests and responses to Registry
13. Voter requests and responses to Blockchain node(s)

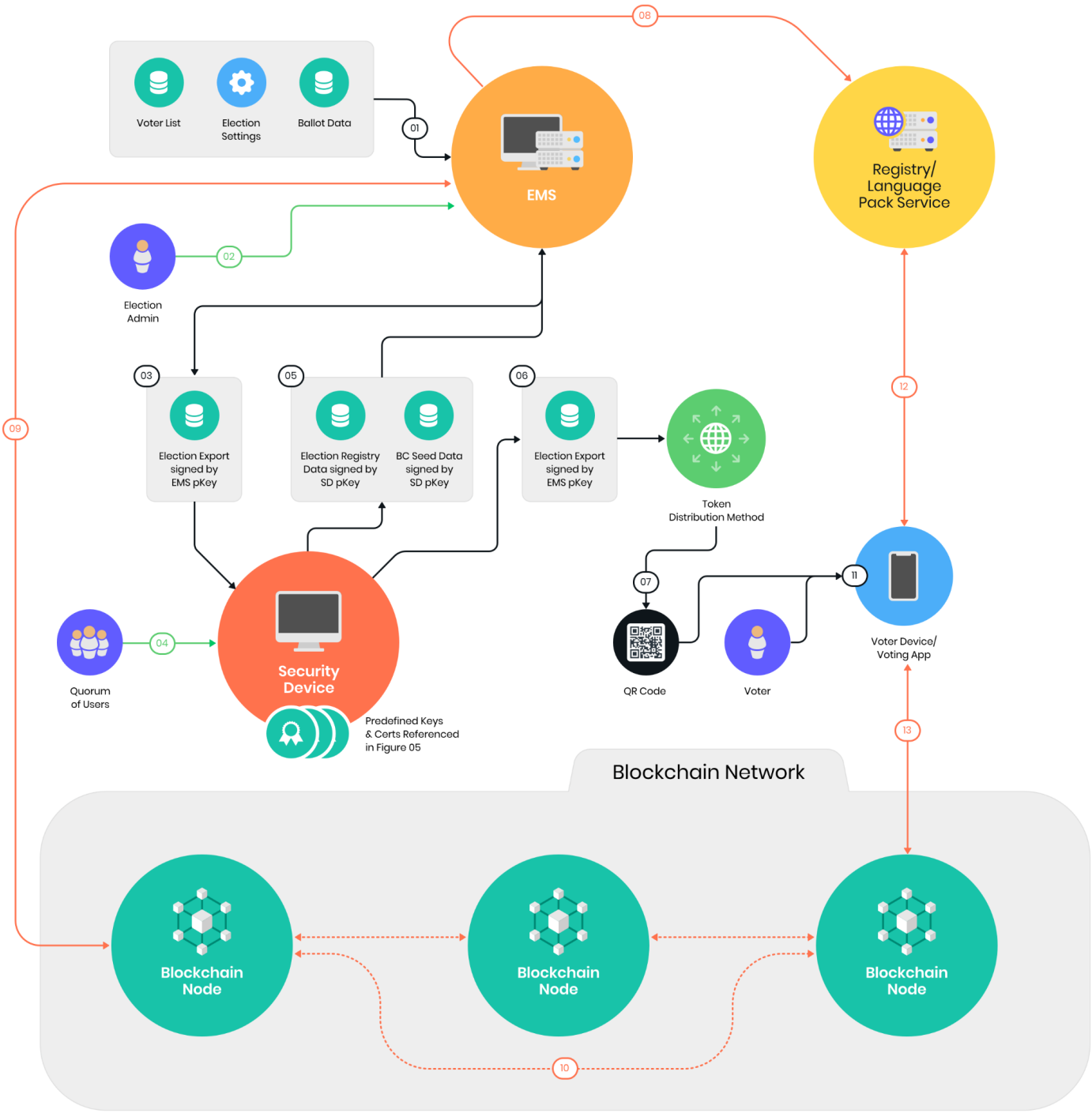

Figure 4: Election Network Detailed Data Workflow

4.6.2 Voter Based Data Flows

The voter interaction with the system is fairly limited in that they will only interact directly with the Voting application on their own mobile device and if they choose to do so, the Receipt Verification application on any other device. All of these data transactions happen during the live election period only with the exception of the last two steps for verifying submitted votes on the blockchain. These actions can be done by the voter at any time after a successful ballot submission through as long as the blockchain remains up. (See *Section 5* for more details)

Workflow Steps (See *Figure 5*)

1. Voter Scans Token
2. Parsed Token data including token signature used to request election details and language packages from Registry. Token signature verified by Registry
3. Election specific information returned, signed by Registry private key
   a. Blockchain verification request sent in background
   b. Verification response if correct Blockchain and election is live
4. Voter enters prompted credentials
5. Voting app creates Master Key from token data and entered creds and all other needed keys (See *Section 5* for more details)
6. Signed Authentication request sent to blockchain
7. If Auth Request if valid, associated ballot returned
8. Ballot marked/reviewed/prepared for submission including creating receipt process
9. Signed and Encrypted Ballot Submission with Receipt
10. Successful submission response OR denied submission response if ballot submission transaction denied
11. Voter enters voter known data for decrypting receipt information
12. Voter request to see receipt and response with receipt

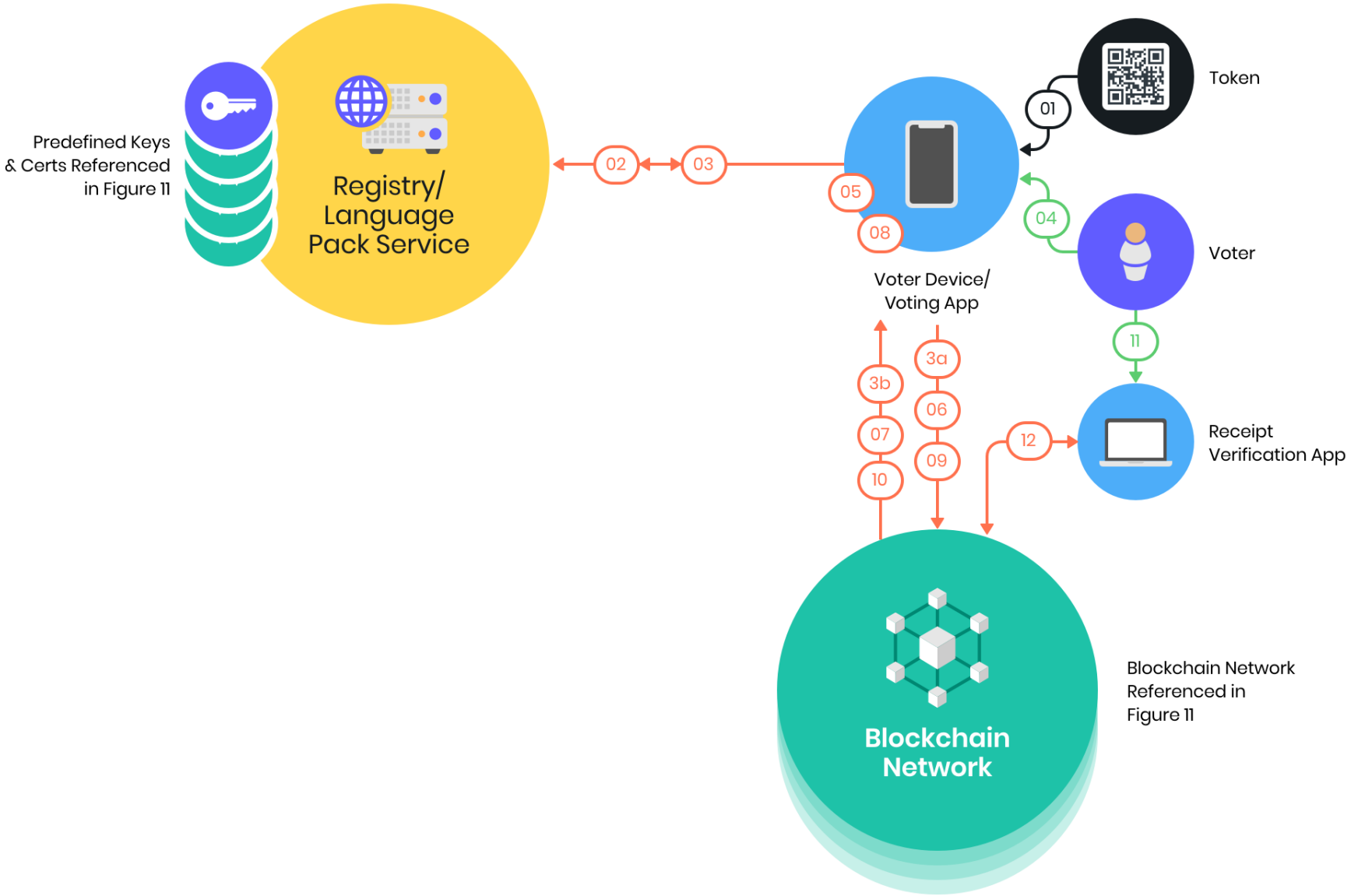

Figure 5: Voter Workflow

4.6.3 Election Administrator based data flows

The election administrator data flows involve many more components than the voter data flows but follow the same general pattern of confirmation between components. All data transfers between components are signed by component private key(s) and verified by the receiving component before a transaction is considered valid. The data flows presented span all stages of the election although there may be slightly different user interface workflows depending on the stage of the election. Example, in pre-election creating an election package to update data on the Registry or Blockchain is relatively straight forward but if a user attempts to update data by first creating the settings/data changes in the EMS during live voting period a series of additional steps are present for the purposes of keeping tight data version controls on voter lists, ballot data, and election settings including supporting languages. This diagram does not indicate the differences in user experience based on such details but rather assumes the transfer methods will be the same even if the on-device user workflows vary slightly.

Workflow Steps (See *Figure 6*)

1. Input by Election Administrator to election data/settings in EMS
2. Election export package created and signed. by EMS private key for delivery to Security Device by secure USB drive
3. Individual or quorum user actions initiated on Security Device
4. Election package(s) signed by Security Device for delivery to EMS by secure USB drive
5. Token package signed by Security Device private key for delivery to Token Distribution Manager
6. Tokens produced and delivered to voters as necessary
7. Registry data package signed by EMS (pass through from Security Device) sent to Registry
8. Registry health checks sent to EMS
9. Blockchain data package signed by EMS (pass through from Security Device) sent to Blockchain network
10. Blockchain node health checks sent to EMS from each node

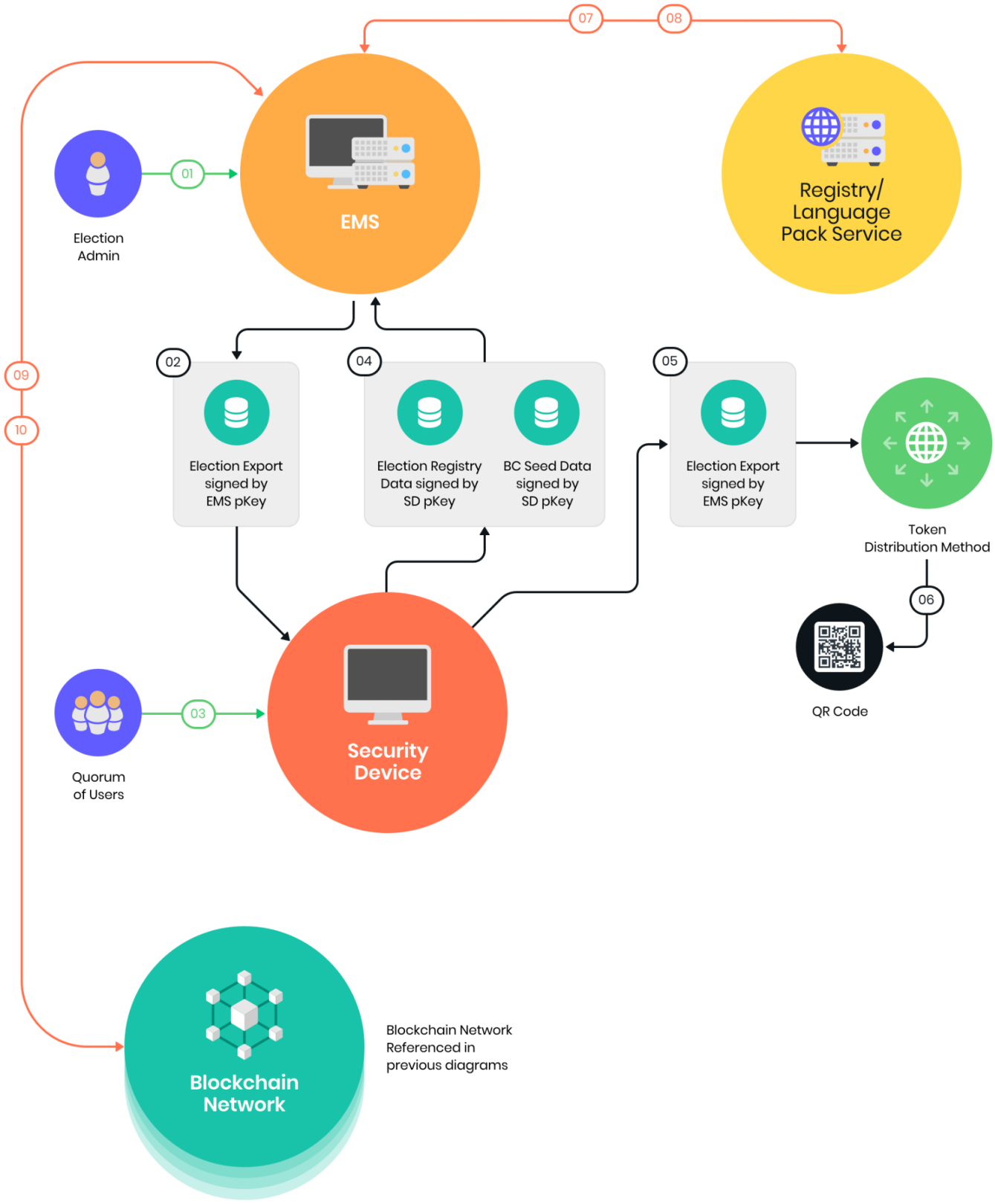

Figure 6: EA Workflows

5 End-to-End Verifiable Proofs

Up to this point the focus has been on establishing the election network and verifying the authenticity and integrity of data transfers between components. While an essential piece to establishing the integrity of the election, by itself these steps do not constitute end-to-end verifiability. To complete this process three verifications must exist. Due to the architecture described below for handling cast vote records (CVRs) the three verifications (Cast as Intended, Recorded as Cast, and Tallied as Recorded) are somewhat overlapping. As such, a review of the basic structure is provided with verification explanations following.

5.1 Vote Submission and Storage Architecture

The CVR collection and subsequent validations require two aspects:

1. **A Voter Receipt** - This receipt contains data that fully encapsulates the ballot choices as well as some user defined data. It is verified against the cast vote record (CVR) and is recognizable to the voter. The receipt is generated on the voting device during the marking process and delivered to the voter at the time of ballot submission. This same receipt is created and stored as a child address (see below) of the cast vote record within the ledger. In order for the marked ballot to be counted later during the tally process the system must verify that the receipt accurately represents the contest choices during post election verification.

2. **Hierarchical Deterministic Addressing Protocol (HDAP)** - The system uses hierarchical deterministic key generation to create a protocol for storing election data on a blockchain network. The protocol provides predetermined locations for election records such as CVRs and associated receipts. The addresses can be traversed from parent to child, but not from child to parent. In this manner a receipt holder verifies the receipt they have in hand cryptographically matches the receipt derived from a CVR stored within the ledger without needing to actually access the CVR, which helps mitigate coercion attacks against voters.

5.1.1 Using Hierarchical Deterministic Addressing Protocol (HDAP)

Vidaloop VotingApp addressing protocol uses hierarchical deterministic key generation based on BIP32[24] to lay out a structure for storing election data in a way that supports end-to-end verification while maintaining the privacy of the voters.

Hierarchical deterministic key generation provides the ability to derive child public key addresses from a parent public key and child private keys from parent private keys. Deriving private keys from public keys is not possible and deriving parents from children is not possible.

The address structure supports the following address types (see Address Hierarchy Figure below):

- **Authentication Address** - Pre-loaded records for each voter that map to a ballot style. By presenting the system with proof of ownership of the private key, a voter is authenticated and delivered a ballot. Poll book check-in transactions can also be posted against this address if the system is configured to do so.
- **Cast Permission Address** - Pre-loaded for each voter who is allowed to submit a cast vote record. This address is a parent for all CVRs for a single voter in an election. This record is separate from the authentication address so that we can support post election reconciliation without exposing a user's ballot selections.
- **CVR Address** - CVRs are children of the cast permission address and store the cast vote records.
- **Receipt Address** - Child of the CVR.
- **CVR Result Address** - Child of the receipt address that stores a document that gives details about the final result of the CVR.

- **Spoiled** - Initiated by the voter if they choose to retract their vote, either to vote again via the voting app or by other means. A spoiled vote is not counted in the results but the receipt and data verification are still processed.
- **Accepted** - The receipt and data verification passed and the CVR is included in the final vote count. (Addressed in section 5.1.6 Post-Election System Receipt Verification)
- **Withdrawn** - The vote is withdrawn by the system during reconciliation because a ballot for this voter was cast by alternate means such as mail or a polling place. (See Section 5.2.2 Additional Notes)
- **Rejected** - The receipt or data verification failed. This state merits a full review of the voting system as it should not be possible under normal circumstances.

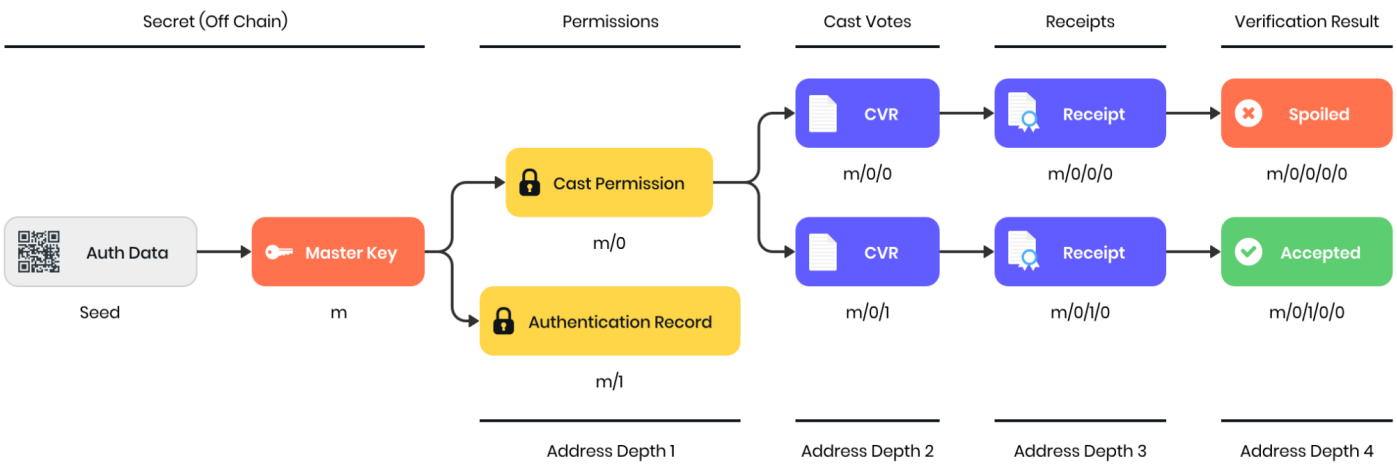

Figure 7: Address Hierarchy

5.1.2 Creating the Addresses

One of the types of data created for the system is authentication data, stored in the public ledger in the form of a public key tied to a ballot style. This public key was first created by the Security Device from the data provided by the EMS, such as a voter credentials detail, a random bit of data generated by the system and later output as the voter token, and the associated ballot style for that voter. On the Security

Device the PPK for each voter is created and will be called the ‘Master Key’ from the diagram above.

From that Master Key two child address keys are created: The Cast Permission and the Authentication Record, both of which come with an associated ballot style ID. In the case of the Auth Record the ballot style ID serves to deliver the correct ballot to the voter, and in the Cast Permission the ballot style ID serves to validate the expected ballot style, which is returned as the CVR. The Master Key is never removed from the Security Device and the private key part of the Master Key is destroyed and never stored. Only the public key remains in the Security Device for the purpose of vote reconciliation.

The public ledger is pre-populated with the list of Cast Permission and Authentication Record at the appropriate addresses..

5.1.3 Authenticating and Casting a Vote

Authentication

The voting application, as mentioned in Section 4, uses the same seed data (predetermined voter credentials and random token data provided in the form of aQR code delivered to the voter separately) and key generation library as the Security Device did to ‘recreate’ the same Master Key, child Cast Permission and Authentication Record keys. For authentication the voting application sends an authentication request transaction to the public ledger signed by the private Authentication Record key. If the ledger can verify a corresponding public Authentication Record successfully it will return the ballot associated with the attached ballot style ID. This Authentication Record and associated transactions stored in the public ledger contain no voter identifiable information but can later be returned to the Security Device, which still contains the public Master Key, for the purposes of producing a list of voters who have checked-in or authenticated in the election.

Casting a Vote

In a very similar manner as authentication, casting a vote from the voting application uses the same Master Key but this time uses the Cast Permission child to create a CVR child and associated Receipt child address. The CVR and associated Receipt child address and data are sent as a transaction to the public ledger signed by the Cast Permission private key. The public ledger confirms said transaction is a valid request by comparing the signature against its list of public Cast Permissions keys/addresses. If one matches it validates two bits of data, that the associated ballot style is as expected and that another CVR and receipt combo haven't already been collected and stored. If a CVR is already present, it will return a response to ask the voter to 'Spoil' the previous submission. Spoiling involves the application creating another child address from the previous receipt address to indicate the intention to not count that CVR. The CVR never leaves the public ledger but the additional spoil address in the hierarchy can be used later to remove that collected CVR from tally operations. The frequency and number of spoils allowed is configurable to meet jurisdiction preferences.

5.1.4 Creating the Receipt

The receipt will have the following data elements:

- **Ledger Address -** Location where the receipt data is stored and can be used to look it up from any device.
- **User Defined Code -** A code or password that can be defined by the user**.**
- **CVR Digest -** Full digest of the contest selections, user defined code and additional optional entropy.
- **Receipt Code -** Version of the CVR digest that is formatted for easier recognition by a human.

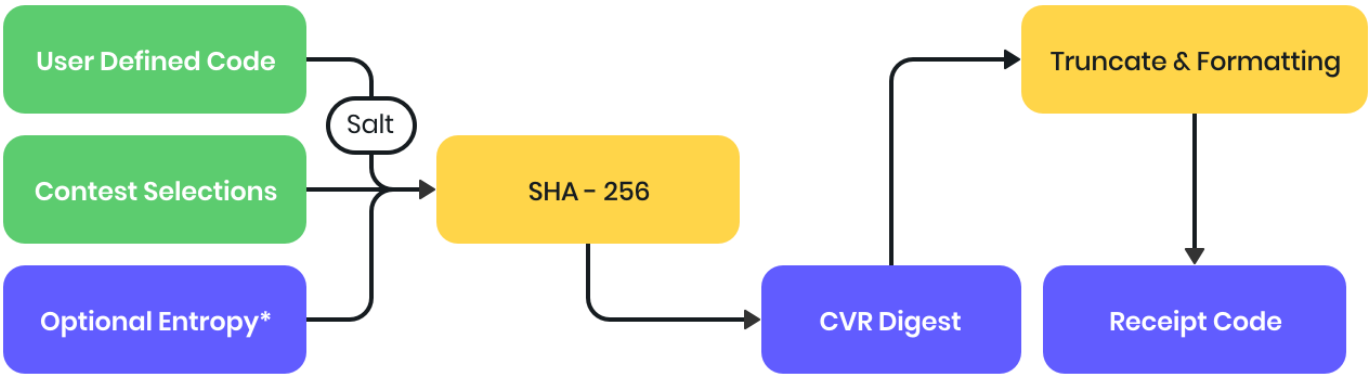

Figure 8: Receipt Structure

The CVR digest and receipt code is generated by creating a digest and truncated digest from the relevant data. This receipt is offered to the voter for storage/printing by the voting device at the time of creation & prior to submission to the public ledger. To increase coercion resistance,the saved/printed receipt has details encrypted by prompting the user for a password or pin.

5.1.5 Voter Validation

It is important that a technically sophisticated voter can recreate the CVR Digest on their own to validate the accuracy of the receipt. In order to facilitate this the Voting Client allows a user to see a detailed view that lists all of the vote selections used to generate the receipt code as well as an explanation on the digest generation. With this data the user recreates the digest and confirms the receipt is constructed correctly.

Less technically inclined voters still get peace of mind by observing that their receipt code changes in a deterministic manner as they vote because the receipt code displays on all ballot marking and review screens of the Voting Client and changes in real time as they alter selections or change the User Defined Code.

During ballot marking the voter has the opportunity within the UI of the application to:

1. Set and change the voter code (user defined code)
2. Show the receipt code in real time and view the changes as they apply or remove selections
3. Be given informational tabs/call outs that explain the basics of the receipt code generation and why it changes as selections change
4. Be given a detailed informational page with descriptions on how to recreate the code independently for more technically inclined users
5. Provided the receipt in a printable/savable format so that they can compare this receipt against the one saved on the blockchain with their encrypted CVR

After submission all voters can confirm that the correct receipt is stored in the public ledger and matches the receipt generated on their device.

However, due to the encryption of the data on the voter's saved/printed receipt the voter will need to use an open source tool to enter the details known only by them (password/pin) and the data on the hardcopy receipt to successfully reverse the Receipt Address and User Defined Code on the actual receipt stored in the public ledger. Once these details are decrypted a voter can use them to verify the ledger contains the same receipt as they have in hand. This extra level of encryption on the saved/printed receipt can be made optional either by the jurisdiction if they do not want the additional level of coercion resistance due to the extra step for their voters, or by voters on a submission by submission basis if so allowed by the election settings.

5.1.6 Post-Election Receipt Verification

After the election is complete and the CVRs are decrypted, the receipts are verified by the election administrator by recreating the CVR digest and comparing it to the stored receipt. Once verification is complete a record is stored marking the CVR as accepted. Any CVRs found generating receipts that do not match saved receipts are flagged and steps are taken to deal with improperly collected data including invalidating the election if deemed necessary.

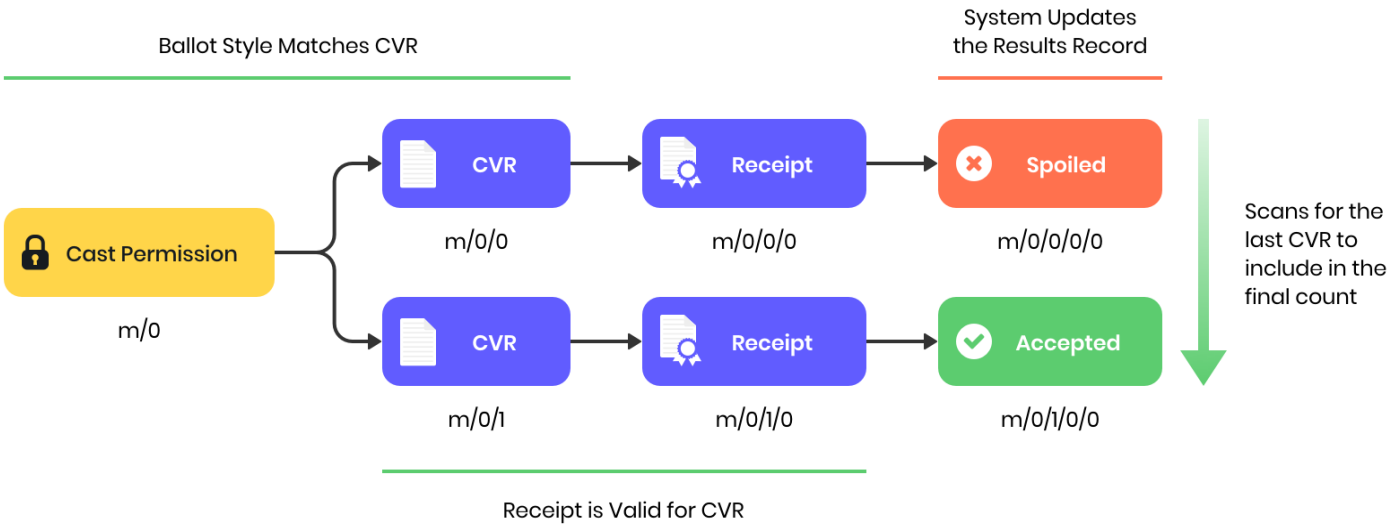

Figure 9: CVR and Receipt Verification Diagram

5.2 E2EV Proofs

5.2.1 Cast as Intended Verification

The Cast as Intended Proof uses multiple steps from the process listed above.

- Voter creates and stores the receipt and receipt code generated by the Voting Client on their own device.
- Voters can confirm that the correct receipt is stored at any time.
- When the CVR is decrypted post-election the voter is assured the Cast as Intended Verification completed because:
  - The receipt stored in the ledger matches the receipt in hand; and
  - The receipt stored in the ledger matches the CVR digest stored in the ledger.

5.2.2 Recorded as Cast Proof

- CVR is decrypted and digest generated.
- Receipt is recreated from above digest and compared against stored receipt, if
  - Receipts match and no spoil child is present CVR is deemed 'Accepted'
  - Receipts that do not match CVR are deemed invalid. Investigation can determine 'why' discrepancy exists and steps can be taken to:
    - Invalidate individual CVR (in case where single instance root cause can be found) and child address of 'Rejected' added; or

- Invalidate entire election because no root cause can be found, or systematic tampering/failure cannot be ruled out

Note: Using the Security Device with additional data (a list of voters who voted in other methods such as VBM, in-person, etc…) the Security Device could also create a child address for receipts tied to CVRs of voters who voted in other methods. This child would be in the Withdrawn status thus removing the CVR from final tally calculations. Assuming there is more than one voter with a Withdrawn status per ballot style, privacy would not be compromised as the Security Device internal operations would handle the ties between voter ID and status at the Master Key level and only output list of Receipt addresses that need additional Withdrawn child addresses attached.

5.2.3 Tallied as Recorded Proof

Unlike other E2EV systems that keep CVR data encrypted, which then requires mathematical proofs for the correctness of vote counting, Vidaloop VotingApp ends with unencrypted CVRs that can be treated similarly to traditional paper ballot for tallying, reporting, auditing, and recounts. Assuming the previous two proofs are successfully validated, a jurisdiction can release the CVR data the same way ballot images and CVR data are released in traditional paper systems.

6 Conclusions and Future Work

A main advantage of Vidaloop VotingApp E2EV is that it ultimately leaves the cast vote record used for tabulation purposes in a human readable and auditable format without compromising voter privacy. The use of an HDAP allows verification by different entities to occur cryptographically without necessarily giving up all the information to reveal identity. The system utilizes a purpose built hardware security device to manage much of the key generation and privacy maintaining functions, but doing so allows the rest of the system to be fully publicly visible and auditable during all stages of the election cycle.

Areas not addressed by this white paper but are important pieces to consider if applying this system to a real life implementation scenario are in no particular order:

1. The detection and prevention of malware on the end user (voter) devices.
2. Details on physical security and process security controls for the Security Device, the public ledger network, the registry, and the EMS.
3. Details on the Token Management System methods and related security concerns.
    a. Paper token delivery has a different set of security concerns than an electronic delivery to the voter.
4. Details on updating voter lists and election data including settings and ballot data during live voting period.
5. Network deployment scenarios related to scalability of the system.
6. How to encourage/ensure enough voters follow through with the full Cast as Intended Proof and verify their receipts against Accepted receipts such that we can create a statistically significant confidence interval in the accuracy and integrity of the election that meets industry standards. Theoretically every voter can follow through but realistically many won't and if no voters validate their receipts against Accepted CVRs then we cannot be sure as election administrators or observers that the Proof of Cast as Intended holds true. There is no current mechanism proposed for encouraging participation or even what levels of participation are needed.

7 References

1. Hastings, N., Peralta, R., Popoveniuc, S., Regenscheid, A. (2011). "*Security Considerations for Remote Electronic UOCAVA Voting*". Accessed July 15, 2021. https://csrc.nist.gov/publications/detail/nistir/7770/final
2. Regenscheid, A., Beier, G. (2011). "*Security Best Practices for the Electronic Transmission of Election Materials for UOCAVA Voters*" Accessed July 15, 2021. https://csrc.nist.gov/publications/detail/nistir/7711/final
3. Gharadaghy, R., Volkamer, M. "*Verifiability in Electronic Voting Explanations for Non Security Experts*" (PDF). Retrieved July 10, 2021. http://subs.emis.de/LNI/Proceedings/Proceedings167/151.pdf
4. Beroggi, G. (2008). "*Secure and Easy Internet Voting*" (PDF). Retrieved July 15, 2021. https://sargasso.nl/wp-content/uploads/2008/07/internetvoting.pdf
5. Rubin, A.D. (2001). "*Security Considerations for Remote Electronic Voting over the Internet*" (PDF). Retrieved July 15, 2021. www.cs.jhu.edu/~rubin/courses/sp03/papers/e-voting.security.pdf
6. U.S. Election Assistance Commission (2011) "*A Survey of Internet Voting*" (PDF). Retrieved July 21, 2021. https://www.eac.gov/sites/default/files/eac_assets/1/28/SIV-FINAL.pdf
7. Kelleher, W. (2013) "*Internet Voting in the USA*: *History and Prospects*" (PDF). Retrieved July 21, 2021. https://www.eac.gov/sites/default/files/eac_assets/1/28/William-Kelleher-Internet-Voting-WPSA-Paper-July-9th.pdf
8. Alvarez, R.M., Hall, T.E. (2004). "*Point, click & vote*: *the future of Internet voting*". Brookings Institution Press, Washington D.C.
9. Evans, D., Paul, N. (2004). "*Election Security*: *Perception and Reality*" (PDF). Retrieved July 14, 2021. https://www.academia.edu/1126471/Election_security_Perception_and_reality
10. Moynihan. D.P. (2004). "*Building Secure Elections*: *E-Voting, Security, and Systems Theory*" (PDF). Retrieved July 15, 2021. https://doi.org/10.1111/j.1540-6210.2004.00400.x
11. Ryan, P. Y. A., Peacock, T. (2006) "T*hreat analysis of cryptographic election schemes*" (PDF). Retrieved July 14, 2021. http://vote.cs.gwu.edu/vsrw2006/papers/7.pdf
12. Gillum, J., Huseman, J. "The Iowa Caucuses App Had Another Problem: It Could Have Been Hacked". Retrieved July 13, 2021. https://www.propublica.org/article/the-iowa-caucuses-app-had-another-problem-it-could-have-been-hacked
13. Specter, M., Koppel, J., Weitzner, D. (2020). "*The Ballot is Busted Before the Blockchain*: *A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections*" (PDF). Retrieved July 9, 2021. https://internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf
14. *Election Assistance Commission (2021). "Voluntary Voting System Guidelines VVSG 2.0" (PDF).* Retrieved June 3, 2021. https://www.eac.gov/sites/default/files/TestingCertification/Voluntary_Voting_System_Guidelines_Version_2_0.pdf
15. Benaloh, J., Rivest, R., Ryan, P., Stark, P., Teague, V., & Vora, P. (2015). "*End-to-end verifiability" (PDF).* Retrieved July 9, 2021.

https://escholarship.org/content/qt7c9994dg/qt7c9994dg_noSplash_97d64dc5a809c552701079250f47b4cb.pdf
16. Grassi, P., Fenton, J., Newton, E. (2017) "*NIST Special Publication 800-63B Digital Identity Guidelines*" (PDF). Retrieved July 15, 2021. https://www.nist.gov/itl/applied-cybersecurity/tig/projects/special-publication-800-63
17. Oppliger, R. (2002). "*How to Address the Secure Platform Problem for Remote Internet Voting*" (PDF). Retrieved July 15, 2021. http://www.esecurity.ch/Pubs/sis_2002.pdf
18. Ronald L. Rivest (2006). *"The ThreeBallot Voting System" (PDF). Retrieved June 1, 2021.* http://theory.csail.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf
19. "Punchscan.org". Accessed July 9, 2021. https://web.archive.org/web/20070928140001/http://punchscan.org/
20. Ryan, P.Y.A.; D. Bismark; J. Heather; S. Schneider; Z. Xia (2009). *"The Prêt à Voter Verifiable Election System" (PDF)*. Retrieved June 9, 2021. https://web.archive.org/web/20101128061130/http:/www.pretavoter.com/publications/PretaVoter2010.pdf
21. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R., Ryan, P.Y.A., Sherman, A. (2008). "*Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes*" (PDF). Retrieved June 12, 2021. http://scantegrity.org/papers/scantegrityIEEESP.pdf
22. "*ElectionGuard*". Accessed May 28, 2021. https://www.electionguard.vote/
23. Joaquim, R., Ribeiro, C., Ferreira, P. (2009) "*VeryVote: A Voter Verifiable Code Voting System*" (PDF). Retrieved July 7, 2021. https://link.springer.com/chapter/10.1007/978-3-642-04135-8_7
24. Pieter Wuille "BIP-032 - Hierarchical Deterministic Wallets". https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

Appendix A - Definitions

AES-256-GCM - *Advanced Encryption Standard 256 bit Galois Counter Mode*
A symmetric block cipher.

BIP32
A hierarchical deterministic wallets scheme, which makes it possible to share an extended public key (K, c) between sender and receiver, where K is a public key and c is a 256-bits chain code, and only the receiver knows the corresponding private key of this K.

CSR - *Certificate Signing Request*
A request generated by a system that has created a public/private key pair. The request is for another system that is trusted to sign its certificate so that it can inherit the trust, creating the chain of trust.

EAC - *Election Assistance Commission*
The federal oversight body created by the Help America Vote Act (HAVA) to provide technical guidance and assistance to states with regard to election administration, standards setting and voting system testing.

ECC - *Elliptic Curve Cryptography*
An approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields.

ECDH - *Elliptic-curve Diffie-Hellman*
A key agreement protocol that allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key. The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher.

ECDSA - *Elliptic Curve Digital Signature Algorithm*
Encryption algorithm based on the algebraic properties of modular exponentiation, together with the discrete logarithm problem, which is considered to be computationally intractable. The algorithm uses a key pair consisting of a public key and a private key.

ECIES - *Elliptic Curve Integrated Encryption Scheme*
A hybrid encryption scheme which provides semantic security against an adversary who is allowed to use chosen-plaintext and chosen-ciphertext attacks. The security of the scheme is based on the computational Diffie–Hellman problem.

PKI - *Public Key Infrastructure*
A technology for authenticating users and devices in the digital world by having one or more trusted parties digitally sign documents certifying that a particular cryptographic key belongs to a particular user or device.

secp256k1
secp256k1 refers to the parameters of the elliptic curve used in public-key cryptography. It was constructed in a special non-random way which allows for especially efficient computation.

SHA-256
A hashing algorithm used to convert text of any length into a fixed-size string of 256 bits.

VVSG 2.0 - *Voluntary Voting System Guidelines 2.0*
The latest version of the Election Assistance Commission's (EAC) federal voting system standards. This is the most widely accepted voting system standard in the country although many states have their own standards or additions to the EAC provided standards.

Appendix B

Encryption and Digital Signatures

The system components use common algorithms and encryption schemes to function correctly.

B.1 Encryption and Signing Keys

All keys and certificates in the system are generated and used for a single purpose. Keys will not be reused for a purpose or domain other than how they were originally intended. This means that separate certificates and keys will be used for encrypting and signing data, as an ECDSA key may only be used for message signing and a ECDH key may only be used for key agreement.
See FIPS 186-4

B.2 Asymmetrical Encryption/Decryption

The system will use ECIES (*Elliptic Curve Integrated Encryption Scheme*). The ECIES standard combines ECC-based asymmetric cryptography with symmetric ciphers to provide data encryption by EC private key and decryption by the corresponding EC public key.

The scheme can use various key derivation functions and symmetrical encryption algorithms. We will be using the following in our ECIES encryption:

- **KDF**: HKDF-SHA256
- **Symmetrical Encryption**: AES-256-GCM

B.3 Signing Data

Data will be signed with the signer's private key and verified by the recipient with the public key. Signatures in the system will be ECDSA with SHA-256: the data will be hashed using the SHA-256 algorithm and then signed using ECDSA.

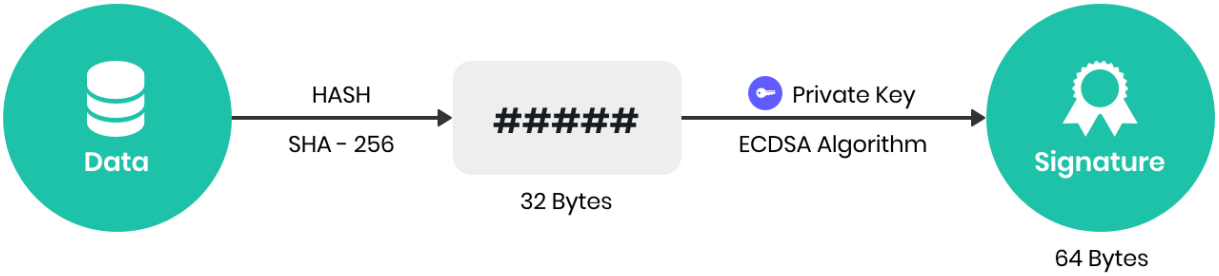

Figure 10: Signing Data

Signer Identity

If the data includes the public key or the data creator provides the public key over a non-secure protocol, validating the signature will only provide limited value, as it is only validating that the data is as the sender sent it, but the sender may be unknown.

In this system the signer must:

1. Be a known and trusted source, or
2. Have a certificate that is derived from a known and trusted source (root of trust)

In most cases the signer must be a known and trusted source as the data sources are known and their certificates are installed at the time the components are provisioned. In cases where the certificate is not installed on the system and verifying the root or signer is acceptable, the public key certificate may be sent with the data and the component will verify the chain of trust to decide whether the data is trusted and can be used or if it should be rejected.

B.4 Encrypting Signed Data

The system will use the *encrypt then sign* method for signing confidential data. This allows all participants to verify the identity and authenticity of the election data even if it is confidential. If the data was signed and then encrypted, the participants would not be able to verify the signature without decrypting the data.

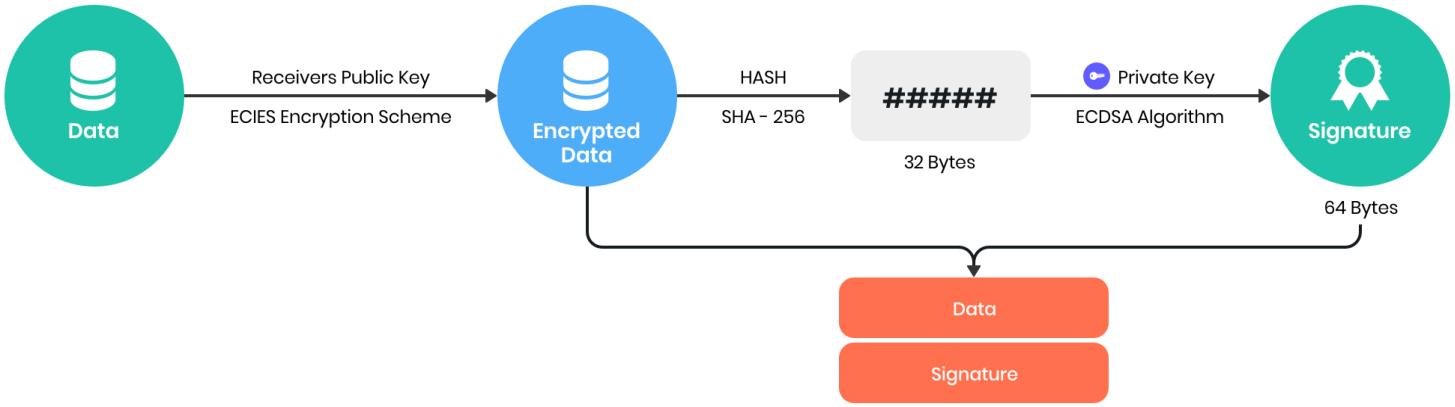

Figure 11: Encrypting Signed Data

Appendix C - Component Provisioning

C.1 Provisioning the Security Device

The Security Device contains a hardware chip capable of producing private keys and storing them inaccessibly for anything other than performing operations. These keys cannot be extracted, tampered with, or manipulated once stored in the hardware chip.

1. During manufacturing of this device the hardware chip is used to create the Security Device Private key and associated Certificate Signing Request.
2. The Security Device certificate is signed by the Vidaloop private key creating a tie between that particular device and Vidaloop.
3. Stored on the device upon delivery to each customer is the Security Device private key unique to that device with a signed Security Device certificate and a copy of the Vidaloop Root Certificate.
4. This same set of keys and certificates is installed on three different devices for the purposes of creating backups should hardware failure render one inoperable.

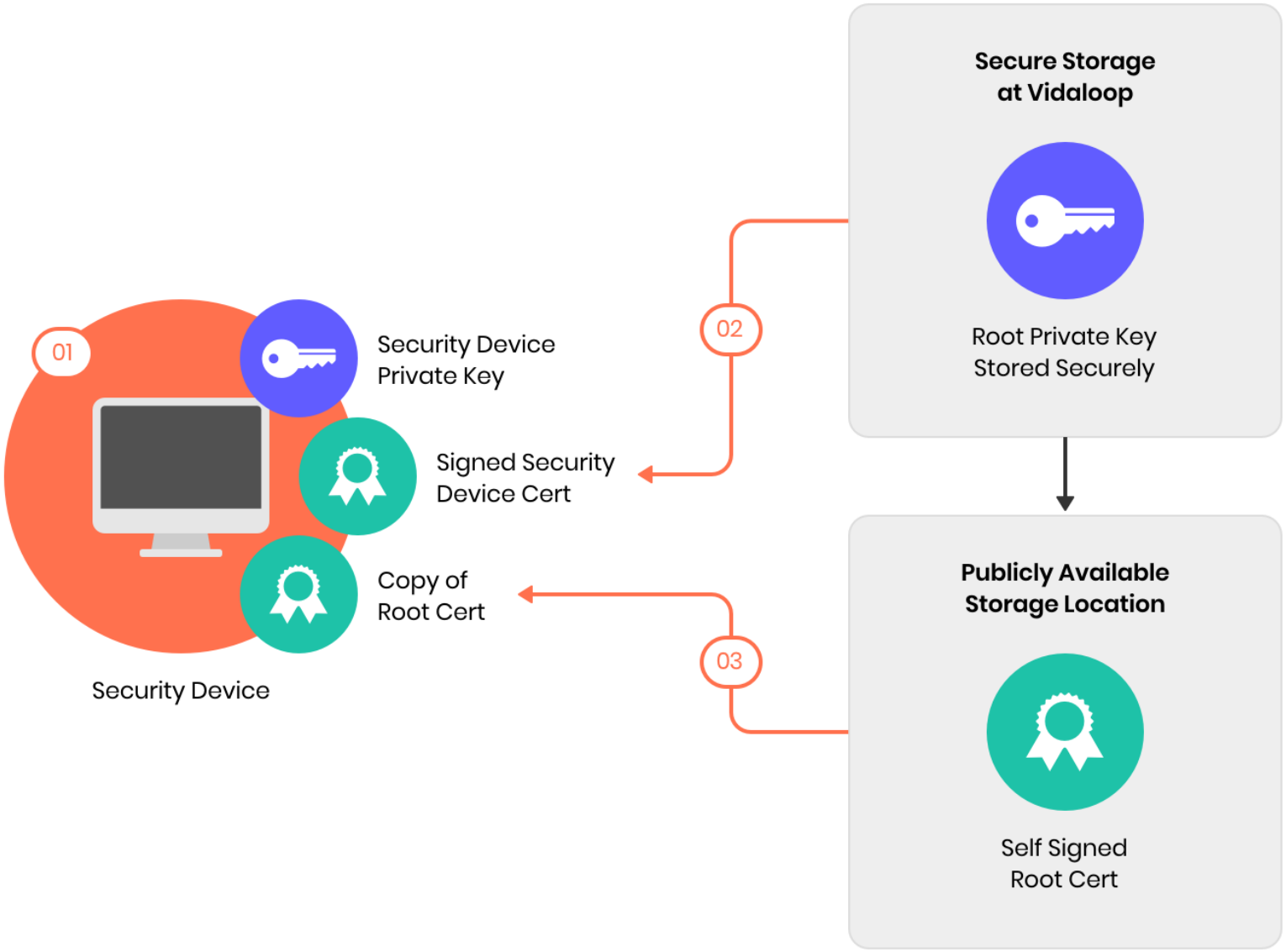

Figure C.1: Provisioning the Security Device

C.2 Provisioning an EMS

The EMS consists of an EMS server and an EMS access device, such as a laptop computer. The server and the access device are connected via a network, which can be a dedicated private network. The EMS and the Security Device will be located in the same office, easily allowing manual data transfers using secure flash drives.

1. The Vidaloop root certificate on the server is installed at code build time, thus providing to the built and deployed EMS code the same tie back to Vidaloop as all other components.
2. The EMS first creates two private keys unique to that EMS deployment. It also creates two EMS Certificate Signing Requests.
3. A data package bundle is created containing both CSRs. The data package is copied onto a secure USB drive and uploaded on to the Security Device.
4. At this point the Security Device will need a quorum of users to initiate the provisioning process to ensure a single individual cannot corrupt this essential process. The Security Device uses its private key to sign the EMS CSRs and keeps a signed copy of the EMS certificate as well.
5. An export with the signed EMS certificates and a copy of the Security Device signed certificate created in the security device provisioning step is transferred to a secure USB drive to be delivered back to the EMS.
6. The EMS uploads the certificates and stores.

Note: Each Security Device can only be registered to a single instance of the EMS at any given time and each EMS instance can only be registered to a single Security Device. In this manner these two critical components become joined in operations from this point forward.

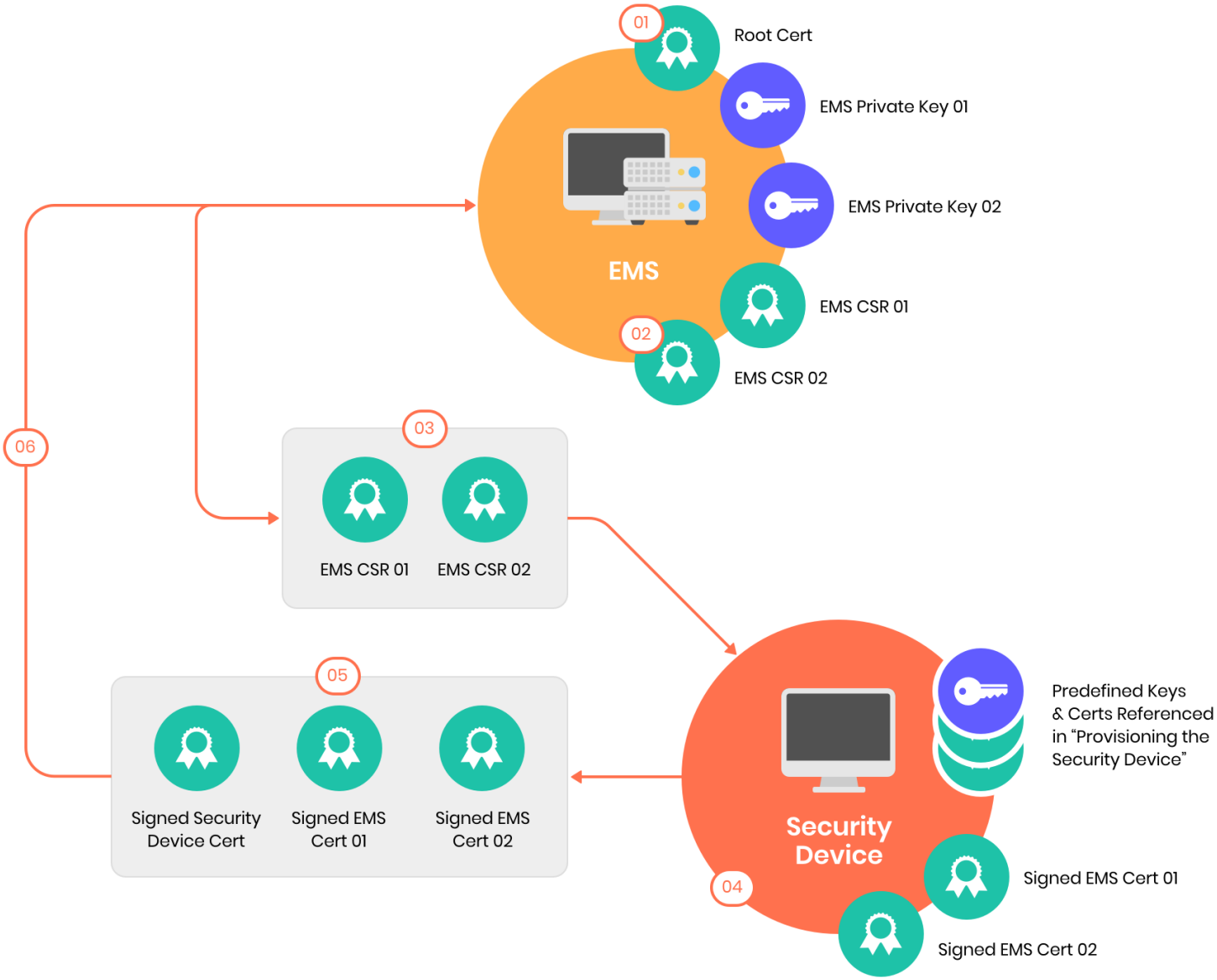

Figure C.2: Provisioning the EMS

C.3 Provisioning the Registry

The registry will be hosted externally, making secure manual data transfer difficult. Therefore, instead of requiring the Registry to create its own key and CSR as the EMS did, the Security Device may generate the private key and certificate (signed by the Security Device).

1. EMS defines Registry settings and creates export package
2. Registry settings package uploaded to Security Device
3. The Security Device creates two different export packages:
    a. A package with the signed Registry certificate to be transferred to the EMS via secure USB drive; and
    b. A package with the Registry private key, Registry signed certificate, copy of Security Device signed certificate, and a copy of the EMS signed certificate, as well as a registry data setup package.
4. Once the Registry operator has installed the second package into the Registry and the first package is uploaded into the EMS, the two components can register with one another by comparing certificate

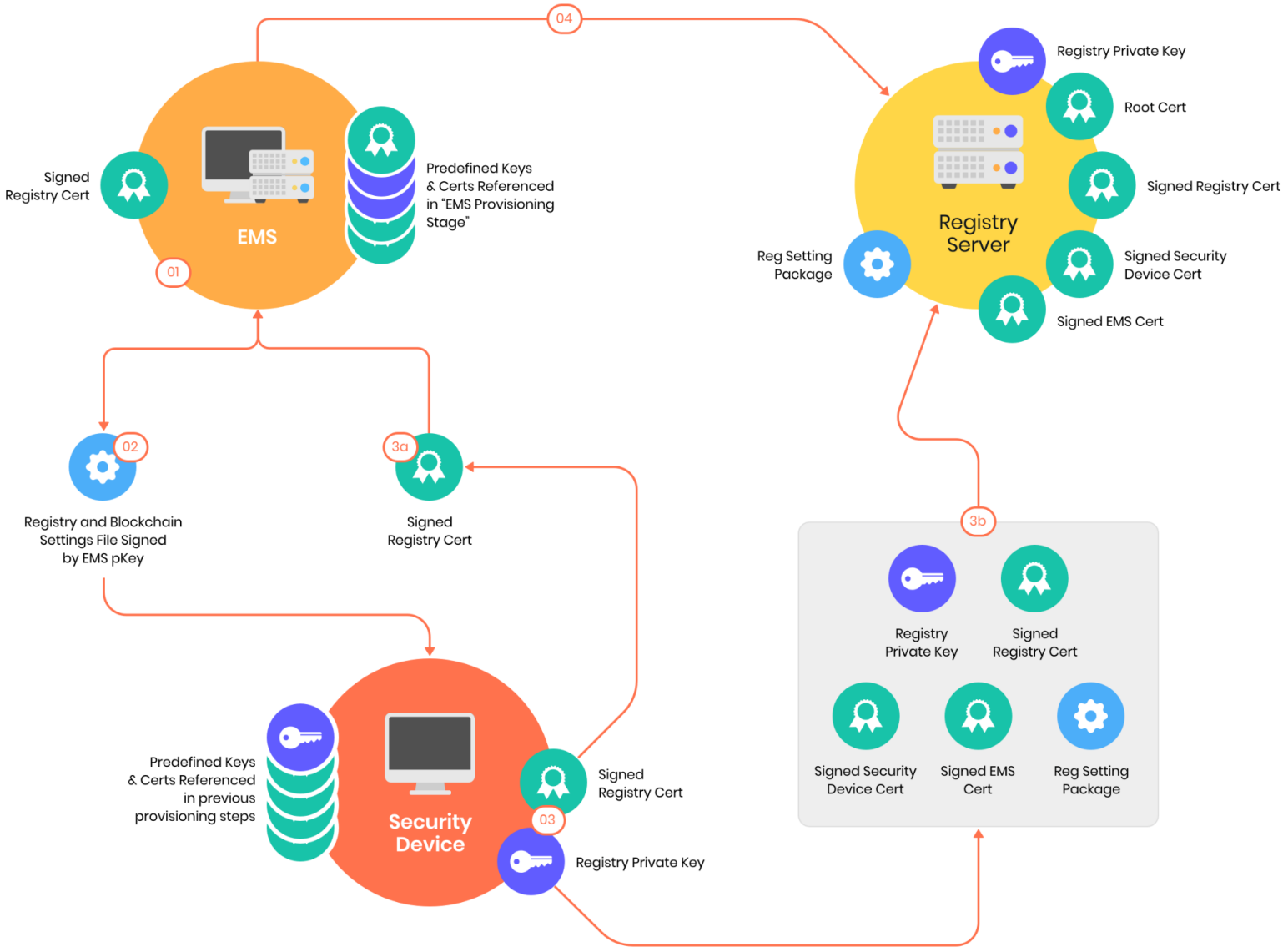

Figure C.3: Provisioning the Registry

C.4 Provisioning the Blockchain

The Blockchain is provisioned in a similar manner as the Registry and is done for each election as one instance of the blockchain is associated with only one election. However, because there is likely more than one Blockchain node operator, instead of a single package sent to the Blockchain, the Security Device will make a package of keys, certificates, and settings for each group of nodes handled by unique operators/trustees. Each Blockchain node will have its own private key and signed certificate created by the Security Device for the purposes of being able to distinguish transactions initiated by individual nodes from other nodes during election operation. Therefore, the EMS will receive a separate signed Blockchain certificate for each node to store within the EMS. Once these are installed on the EMS and each node is operational the EMS can be networked to each node for the purposes of sending election setup data and election updates as needed.

1. EMS creates new blockchain settings file which is exported via secure USB drive to the Security Device
2. Security Device uses settings file to create *n* number of blockchain private keys and certificates signed by the Security Device private key
3. Security Device creates one package with all the blockchain signed certs for delivery via secure USB drive to the EMS for upload
4. Security Device creates multiple packages with appropriate blockchain private keys (number determined by how many nodes each trustee will manage), all blockchain signed certs, EMS signed cert, Security Device signed cert, and Registry signed cert for delivery via secure USB drive to blockchain operators/trustees
5. Blockchain operators install package from Security Device and when all associated certs are also loaded in the EMS, direct connection between EMS and blockchain can be established for the following purposes
    a. Network health checks by EMS to blockchain
    b. Delivery of election data and settings to blockchain from EMS
    c. Delivery of blockchain data (collected CVRs, log data) post election from blockchain to EMS

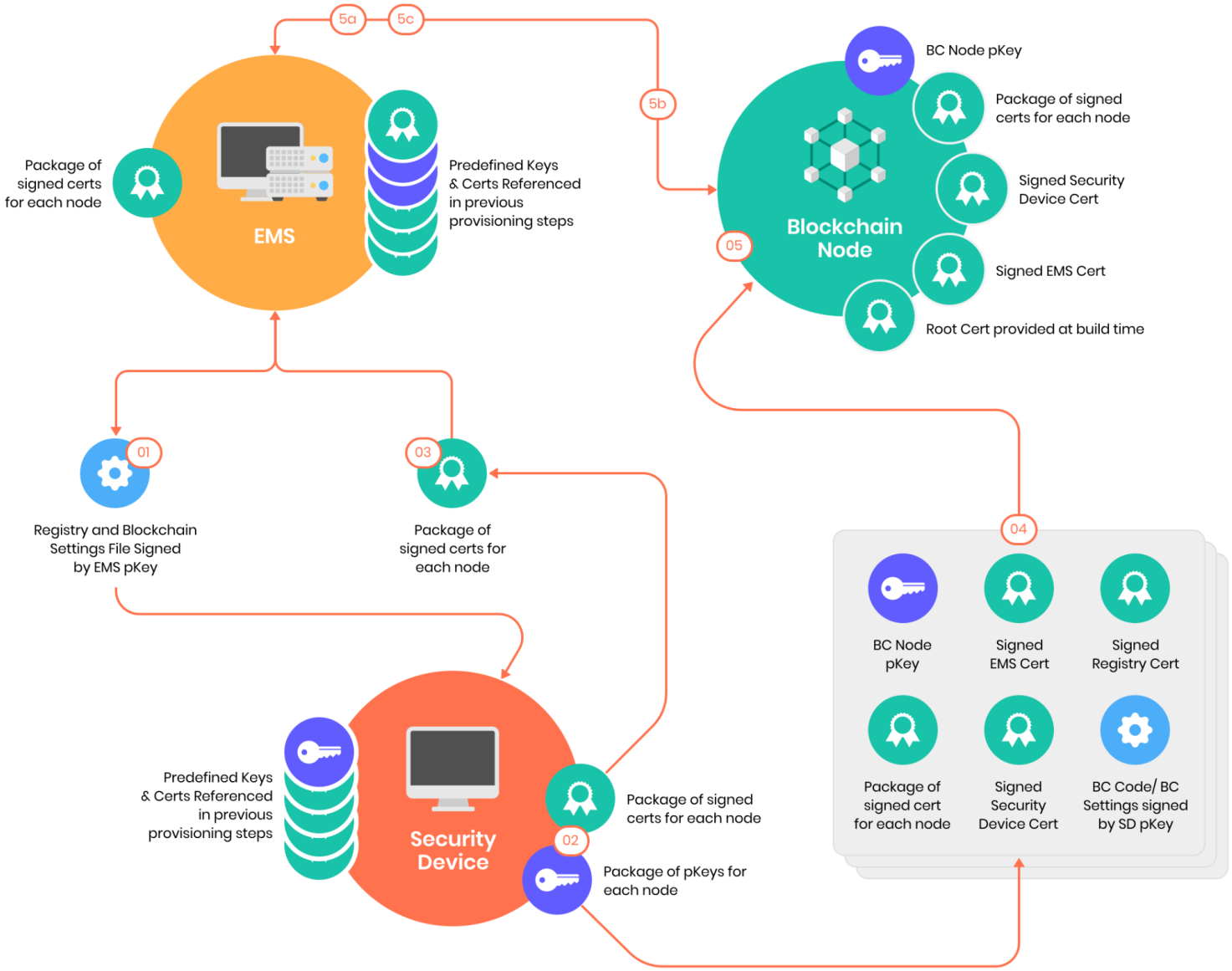

Figure C.4: Provisioning the Blockchain

Appendix D - Technologies used

D.1 Cryptographic conventions

The following is a brief description of the cryptographic conventions used within Vidaloop VotingApp and a brief explanation on the choices made. These are standard cryptographic conventions applied in well vetted and accepted methods.

1. Signing Algorithm: ECDSA with SHA-256
   Elliptic Curve Digital Signature Algorithm (ECDSA) with Secure Hash Algorithm 2 (SHA)-256
2. Key Agreement Protocol: ECDH
   Eliptic-Curve Diffie-Hellman (ECDH)
3. Symmetric Encryption Algorithm: AES-256-GCM
   Advanced Encryption Standard (AES)-256-Galois Counter Mode (GCM)
4. Asymmetric Encryption Scheme: ECIES
   Elliptic Curve Integrated Encryption Scheme (ECIES)
5. Elliptical Curve parameters: secp256k1 (ledger transactions), P-384 (key exchange & signatures)

Elliptic curve cryptography (ECC) was chosen for several reasons:

1. **Common Stack**: It is the technology behind most blockchain systems including hyperledger sawtooth. The use of a common technology stack is preferred whenever the desired results can be achieved while doing so.
2. **Data Size**: *Storage, Bandwidth, etc*: ECC gives a higher level of security over RSA for a given key size. For example it would require a key of 3072 bits in RSA to get the same level of security as a 256 bit ECC key. Smaller keys, certificates, and signatures makes for an overall more efficient system, especially when operating in low bandwidth or embedded conditions.
3. **Flexibility**: RSA works under the same principles as ECIES in that it uses asymmetrical encryption to encrypt a symmetrical key, which is in turn used to encrypt the data. One main difference (other than that mathematics involved) is

the flexibility of the ECIES framework to have different algorithms plugged in to achieve particular results or properties. Various types of curves can be used, each with their own properties. Different key-derivation functions such as PBKDF2 or Scrypt can be used to achieve the results needed in combating attacks. The underlying symmetric encryption algorithm can be chosen to meet the needs of the application. Changing or updating any part of the system is relatively easy from a systems and programming point of view, as is versioning data so the system can transition over time to new and more secure algorithms as needed.

4. **Security**. RSA is an older technology, created in the mid 1970s and as such has more known vulnerabilities. This has been mitigated over time by introducing more secure symmetrical algorithms and larger key sizes. RSA with 1024 bit keys has been invalid due to the lack of security since 2010 and RSA with 2048 bit keys will be obsolete by 2030. In order to have 256 bit symmetrical encryption with RSA it would require a 15360 bit key which is computationally infeasible for most systems. With ECC it is possible to use an underlying 256 bit symmetrical encryption with a standard secp256k1 key.

Use in Government

ECC is approved for use in government systems and is used by the NSA to protect data up to the Top Secret level. The NSA has been swaying standards bodies away from RSA since the late 1990s, and now they have publicly announced that they intend to phase out ECC because of a possible threat from quantum computing. However, they have not provided an alternative recommendation. Currently ECDSA and ECDH, along with AES-256, are still used in the NSA security algorithm suite.

ECDSA is approved within FIPS 186 (Specified in the American National Standard, ANS X9.62)

ECDH is FIPS140-2 compliant using the same curves approved for ECDSA, See NIST 800-175b, section 3.3.4. Key agreement standards are described in ANS X9.63

D.2 Security certificates

Vidaloop VotingApp uses a series of certificate chains between components for the purpose of determining trust in data transferred between components. Each component has a direct certificate tie back to a Vidaloop Root Certificate and most data passed is signed by more than one component thus creating situations where to compromise the data within the system a bad actor would need to compromise multiple components, deployed in distinct geographic locations and maintained by a variety of trustees. In the provisioning portion of this paper the exact location of each certificate and its purpose is identified. See Section 4.4.

Certificate Authority

Digital certificates are issues by a certificate authority (CA). The certificate certifies the ownership of a public key. The CA acts as a trusted third party. This allows others to trust the private key which is paired with a certified public key.

Public Key Certificate

Ownership of a public key is proven by a public key certificate, which contains information about the owner of the key, information about the key, and a digital signature of the issuer, verifying the certificate's contents.

Root certificate

A root certificate is a public key certificate that identifies a root certificate authority. Root certificates are self-signed and form the basis of a public key infrastructure. It is the top certificate in the certificate hierarchy. Its private key is used to sign other certificates in the system, which in turn inherit the trustworthiness of the root.

Intermediate certificate

An intermediate certificate is signed by a root private key or another intermediate certificate, allowing it, in turn, to sign other certificates down the chain.

D.3 Immutable Data Storage

Blockchain is used for one component of the system because of the immutable nature of the blockchain data. With Vidaloop VotingApp data pieces deemed particularly sensitive especially with regard to audit purposes are stored on the blockchain. These include voter authentication data, ballot data, some election settings, and the Cast Vote Records and associated ballot receipts collected from the voter devices upon ballot submission. However, the solution does not depend specifically on blockchain. Any immutable data storage system could be substituted in its place. The characteristics of a blockchain component are what become important in the solution.