Improving Election System Cybersecurity

Ray Lutz, 2022-11-14 raylutz@citizensoversight.org

Updated 2023-01-14 -- added more detail regarding RATS
Updated 2023-06-07 -- version appropriate as VVSG 2.0 Comment
REF: M1999

1. Introduction

Security of voting systems is a difficult problem because the goal is to produce a system that cannot be hacked, does not rely on the trustworthiness of anyone including election officials, can preserve the option for voters to vote privately and make it infeasible to link the identity of the voter to their ballot, and yet provide for full transparency so the public can check the results, and thereby maintain confidence in the outcome.

After the year 2000 election, the Help America Vote Act (HAVA)[1] was enacted to avoid the hanging chad and lack of confidence that surrounded that election. Election equipment was "upgraded" but unfortunately, the first "improved" equipment utilized Direct Recording Electronic (DRE) designs that had no durable paper record, and thus could lose votes. Also the vote count could be easily changed because there was no audit trail. Since that time, the use of a durable paper record is now recognized as essential for ease of voting, for software independence[2] and to allow audits of the results. Further, hand-marked paper ballots excel in the ability to ensure the voter has verified their vote (since they marked the ballot) and the public can review the record to verify that the voter was shown all the contests and options in their private voting session. Further, these ballot designs are also compatible with voting at home or by overseas military or civilian voters.

Today, nearly all jurisdictions also create ballot images that can be reviewed by automated auditing platforms to verify the tabulation by voting systems. Even though ballot images are the first digital representation, and by evaluating the election starting with those images nearly all hacking scenarios of the tabulation can be detected, there still exists a small chance that the images themselves will be modified by malicious fraudsters[3].

[1] https://www.eac.gov/about_the_eac/help_america_vote_act.aspx
[2] Rivest, R., Wack, J. -- "On the notion of "Software Independence" in voting systems" -- https://people.csail.mit.edu/rivest/RivestWack-OnTheNotionOfSoftwareIndependenceInVotingSystems.pdf
[3] Bernard, M., et al -- Unclear Ballot: Automated Ballot Image Manipulation, https://mbernhard.com/papers/unclearballot.pdf ;
See also Lutz, R. "Critical Review of Unclear Ballot": https://copswiki.org/Common/M1976

The goal of this paper is to present a viable approach for extremely strong security of election data by securing it at the source, while not changing the methodology of voting. This differs from the changes suggested by the current VVSG 2.0 regarding E2E voting systems.

Most importantly, we propose securing ballot images as soon as they are created by a voter-facing ballot scanner at a polling place. A similar scenario may be applied with scanners in a central scanning facility. Both are described below. In both cases, the devices are air-gapped with no Internet connection. The polling place case has no communication except for a configuration/results flash drive, while the central scanning operation may have LAN connectivity to local computing resources, but still no Internet connectivity.

2. Overview Of The Strategy:

To obtain this security, the overall process is as follows. The details of these steps are further discussed in the body of the document. While there are many options regarding key sizes and algorithms used, only one option is used in this example. We anticipate the use of JWK Json Web Keys, and other standards that allow changing the algorithms because they document the algorithms internally.

This example is provided roughly in the order of operation in a sample election. (This scenario is offered as a starting point for a more rigorous definition to be resolved by consensus).

1. **FIPS 140-2 Hardware Security Module**
   To properly secure the images and provide for random shuffling of the ballot images and data, the ballot scanner should include a hardware security module which is compliant with FIPS 140-2. The security module should have the capability to generate private and public key pairs and protect the private key, which should be known only by the device. By using a hardware device, the private key can be generated internal to the device and be unknown to all entities outside that device, including the manufacturer and any election workers, and be infeasible to acquire from the device.

   Typical hardware components are available at a low cost (< $10) that use an internal noise source that can generate truly random numbers that are unpredictable. It should be required that this functionality is implemented in a separate Trusted Execution Environment (TEE) which is separate from the rest of the execution environment. Microprocessors commonly can provide a TEE as a secondary executing CPU in the device so it can operate independently from the larger system.

2. **EMS Must Acquire Public Keys of Scanner Devices**
   When the voting machines are initially configured by the election office, a protocol is utilized to communicate with those machines using removable media. The entire interaction with the scanner devices is encapsulated by the TransGapProtocol (TGP) which can request data from air-gapped devices using removable media.

The first step is to acquire the public key of each machine using the approach specified by NIST Special Publication 800-89 "Recommendation for Obtaining Assurances for Digital Signature Applications"[4]. This protocol utilizes a random number provided by the Election Management System commonly called a 'nonce', short for "number used once". The nonce prevents attackers from replaying previously intercepted authentication messages, and is like a "wet signature" on a contract, because the returned message including the public key has to be generated using the nonce, and the nonce can't be reused.

Here the request to provide the signed public key using the nonce is sent by the EMS to the voting machine scanner using the removable media. The scanner writes its public key to the media device signed again using the private key and the nonce. This provides the public key, and proves that the scanner also has the private key. Ideally, a new private key pair is generated for each election.

3. **Obtain a Symmetric Encryption Key**
During the initial contact by the EMS, the EMS will provide its public key. The ballot scanner will return a randomly selected symmetric encryption key, which is then encrypted using the public key of the EMS. This encryption key can be longer and better than the public-private key pairs, and used in a faster encryption algorithm which uses a shared encryption key. [Because there is only one round-trip, it may be just as good to encrypt using the EMS public key and dispense with this step.]

4. **TransGapProtocol (TGP)**
It is envisioned that the interaction between the EMS and the ballot scanner devices is defined by an air-gapped system communication standard TransGapProtocol, which should be defined for both communicating from a master air-gapped system to controlled air-gapped systems, and also from a master air-gapped system to the outside world.

TGP has a first phase when the controlled device is provisioned while in a secure physical location, and a second phase when the controlled device is deployed to a hostile environment, where it is used, and then later it returns data using the same flash device. During the entire two phase protocol, the same flash drive is used, and it accumulates data as it is used by both the master and the controlled devices.

In the first phase, the flash device is plugged into the "master" TGP system (the EMS) where it is initialized with the request, including 1. request block, 2. the nonce, and 3. the public key of the EMS.

The drive is then plugged into the controlled device, the ballot scanner. It fulfills the request by writing 1. its public key, 2. RATS protocol EATS evidence (to prove the authenticity of the device, see below), 3. (optionally) a randomly selected symmetric encryption key which is encrypted using the EMS public key, and 4. (optionally) other

[4] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-89.pdf

identifying information onto the flash drive, and signs the package. The flash drive is then returned to the master (the EMS), where it reads the public key of the controlled device (the ballot scanner) and verifies that it used the corresponding private key to sign the package. See Appendix 1.

5. **RATS -- Remote Attestation Procedures**
   The initial TGP request will also include a request for additional information from the scanner device, to provide evidence that the scanner is legitimate. This follows the Remote Attestation Procedures (RATs) protocol. During this process, it is also feasible for the TEE hardware core to provide a complete firmware scan and produce a hash of the installed code, which can be later checked with the officially released code to verify that it has not been altered.

6. **Acquiring the Public Keys**
   The result of the initial TGP phase is the set of public keys of all the scanners that are to be deployed in the election. The flash drives are returned and read by the Master TGP device (the EMS) and it reads and validates the data from the devices. TGP includes the element of tracking all the devices to ensure that all devices originally configured are also read.

7. **EMS Publishes Scanner Public Keys**
   The set of public keys of all ballot scanners must be signed by the private key of the EMS and then published by the EMS for auditing purposes. The total size of the data is less than about 1000 bytes from each scanner device, depending on how many images it creates. Publication of this data is essential to prove that each of the images was signed by the protected private key of the scanner in question. The publication of this data set should occur prior to the start of the election.

   To publish this data, the consolidated data will be written to a flash memory device using TGP, but this time the master system is communicating to the outside world.

8. **Configure Controlled Ballot Scanners**
   The next phase, a data request from the master EMS system to the ballot scanners, has three steps: 1) configuration and request data is written to the device by the master, including a new nonce, 2) the controlled scanner devices read and validate the data from the EMS, and 3) result data is written by the scanner devices and then read by the master.

   At this point, only the first step is conducted, to provide additional configuration data to the ballot scanners, if needed, so they can operate in the field. The configuration data written to each flash drive by the "master" EMS system appropriate for each individual scanner machine. The flash drive is mated to that specific machine and cannot be used in any other device. Included in the configuration information is a nonce which is randomly selected for each machine for this phase by the master.

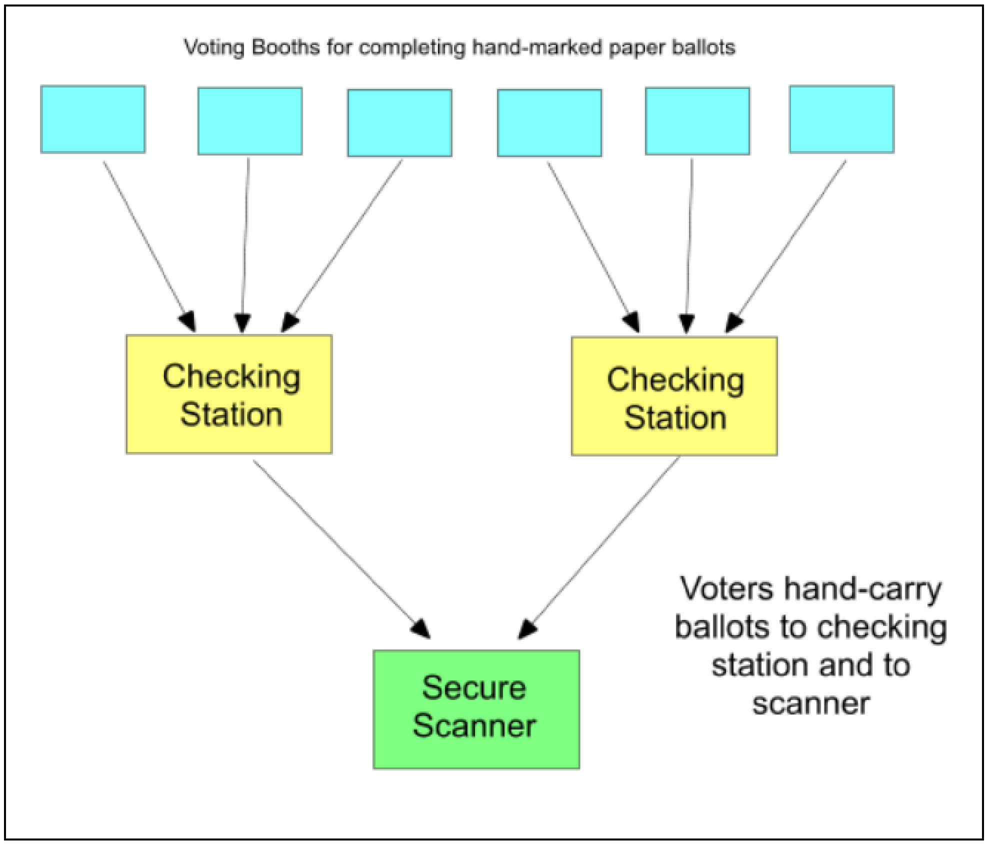
9. **Scanners Deployed**
   The ballot scanners are then deployed to the field (or sometimes in the central office, for central scanning operations) where they operate normally. In the preferred embodiment, each ballot scanner should have two or three internal subsystems:

   a. The Trusted Execution Environment (TEE) which is associated with the security module and can gather evidence about the rest of the system,

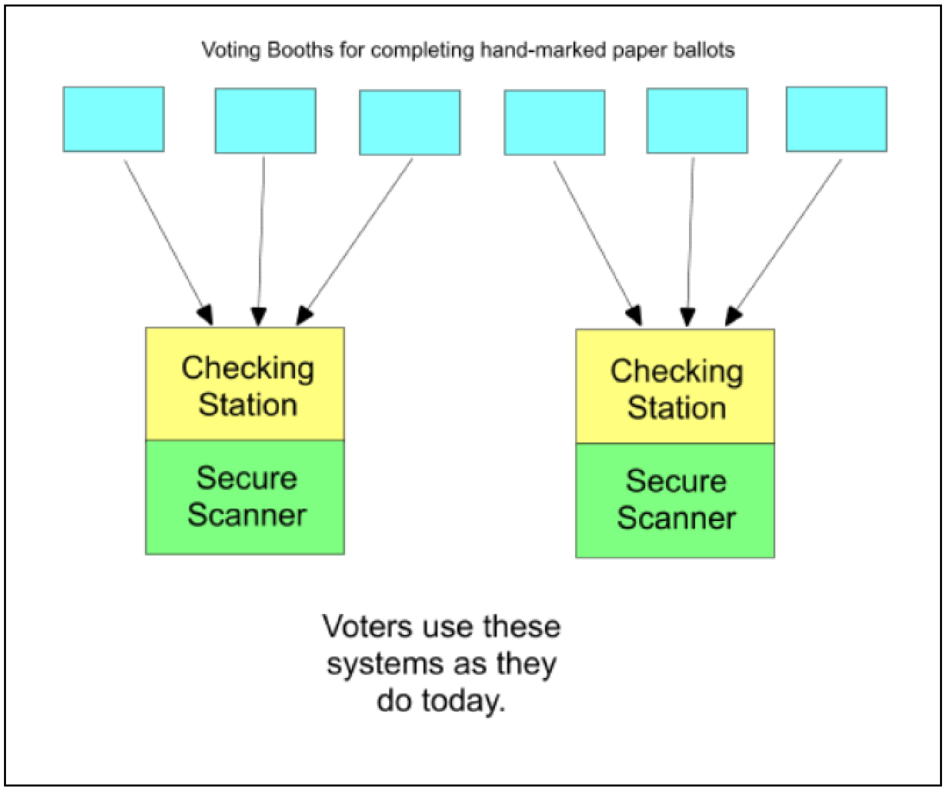
   b. the image scanner element, which can scan images and secure them using the methodology described below, and

   c. the ballot aware tabulator, which can interpret the data from the secured images, and provide error-control feedback to the user and create an informative report of the number of ballots scanned, images created, etc. This last element is optional, but has been a traditional part of voter-facing ballot-aware voting machines. Central scanning operations should not have this part.

   It must be mentioned at this point that there are at least two realizations for the architecture of such scanner devices. In this first and preferred "ballot-unaware secure scanner" realization, voters complete their hand-marked paper ballots in individual private booths using a pen. After completing their ballot, they can insert it in a checking station where their ballot can be checked for overvotes and completely blank ballots (it is uncommon to alert the voter about undervotes and blank contests because many voters do not complete their entire ballot). These checking stations must be aware of the structure of the ballot, but they do not record any votes. Once the voter is confident that their ballot is voted appropriately, it is inserted into the ballot-unaware secure scanner. The secure scanner is unaware of the structure of the ballot and only creates secured images.

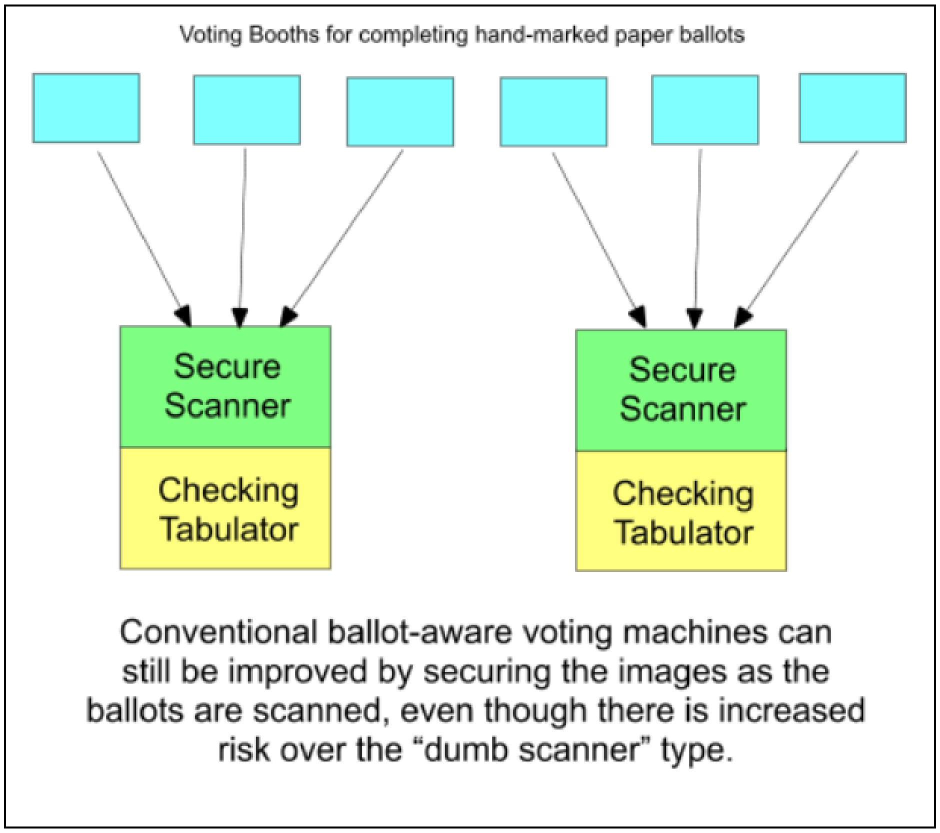

Voting Booths for completing hand-marked paper ballots
Checking Station
Checking Station
Secure Scanner
Voters hand-carry ballots to checking station and to scanner

The second realization is not as secure as the ballot unaware secure scanner architecture, but is similar to what voters are already familiar with. We suggest that the ballot aware scanner is split into two partitions, one which is the checking station, and is ballot aware (but keeps no data of votes), and the second, which is a ballot unaware secure scanner.


Voting Booths for completing hand-marked paper ballots
Checking Station
Secure Scanner
Checking Station
Secure Scanner
Voters use these systems as they do today.

We can note that the combination of the voting booths for hand-marking the paper ballots followed by the checking station is similar to the functionality of a BMD device. For voters with disabilities, it is best if the result of any assistive devices produce a format similar or identical to a hand-marked paper ballot format.

Note that existing ballot-aware scanners can also secure the images, as shown in the diagram below. But if the user decides not to accept the scan due to mistakes such as overvotes, then the tabulator would need to either keep all ballots including the spoiled ones, or allow it to delete some images, which is dangerous from a security standpoint. Also, if the scanner is aware of the semantic content of the ballot enough to check it and tabulate it, then it also has a higher capability of being hacked to modify the image prior to being secured. Therefore, this realization is not recommended, but may still be a step in the right direction in existing ballot-aware voting systems, particularly if the opportunity for installing fraudulent software or firmware is minimal.


Voting Booths for completing hand-marked paper ballots
Secure Scanner
Checking Tabulator
Secure Scanner
Checking Tabulator
Conventional ballot-aware voting machines can still be improved by securing the images as the ballots are scanned, even though there is increased risk over the "dumb scanner" type.

10. **Polling Station Activation**
    Per TGP, the flash drive with deployment configuration is plugged into the controlled scanner device and that device verifies that the new information is from the master by validating the signature of the master. When the ballot scanner is first "opened" for operation, the TEE will collect evidence from the rest of the system regarding a full firmware scan and creation of a hash of the firmware, and collect other evidence to allow the EMS to evaluate the health of the system. This is written using the Entity Attestation Token per the RATS protocol.

11. **Scanning Ballots**
    During the election period, the ballot scanner devices operate identically to how they customarily operate from the perspective of election staff and voters. However, it is critical that the ballot images are immediately secured when scanned and saved to the flash device, as described below.

12. **Ballot Images**
    Image data of each ballot (generally both sides of each sheet) is placed into a standard non-lossy compressed format. The standard core format in use today inside PDF and TIFF files is the CCITT Group 3 facsimile format which compresses each row of bits using Huffman run-length encoding. There is one compressed image created for each

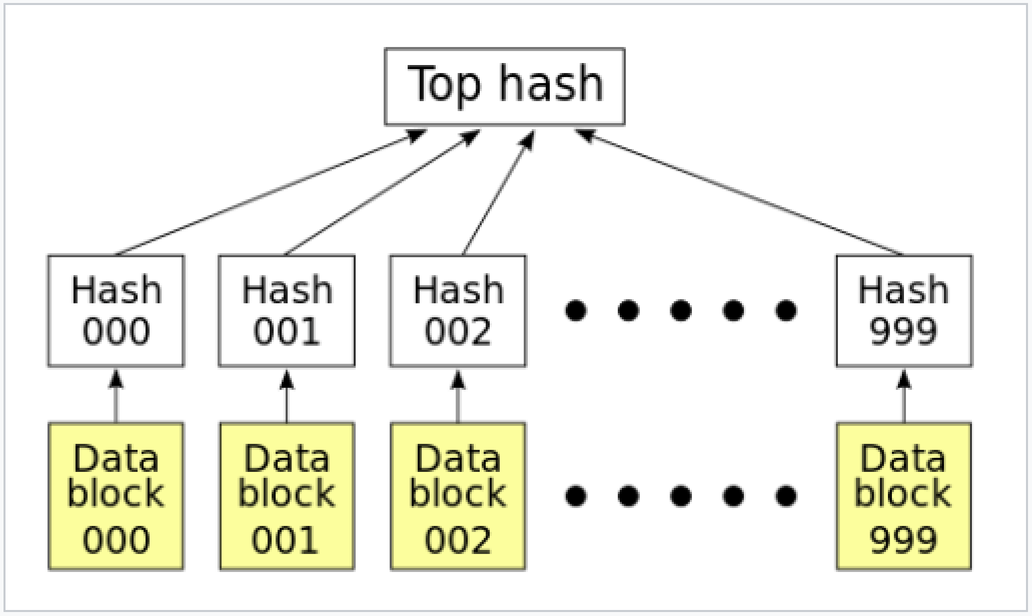
side of the sheet, and these are typically combined into a single file, such as TIFF[5]. The compression varies by complexity of the document image. Typical voting machines today use images of 200 dots per inch (dpi) resolution with 1 bit per pixel, and typically consume perhaps 150K bytes for both sides of a sheet when compressed. But this image size can vary if regions exist that are not easily compressed, and it could be a lot smaller.

13. **Image Hash**
A hash value is generated of this compressed data over both sides using a standard secure hashing algorithm (SHA). It is recommended that SHA-256 is used, generating 256 bits (32 bytes) of data for each pair of images. (Details about how the hashes are created will be deferred for a more detailed specification.)

14. **Sign Each Image Hash**
The scanner will cryptographically sign the hash value that results from each image pair. This is to allow the image to be shuffled to obscure the scanning sequence and thereby to enhance voter privacy, and to allow the image to be signed immediately after creation. This signature can be strong but still small, because the entire set will be signed again. For purposes of this example, assume that a RSA-1024 bit key is used, resulting in 128 bytes of data. We must add this to the 64 bits of hash data that needs to be signed, resulting in about 192 bytes per image. There is a modest amount of additional data required to provide the JSON wrapper. There is no timestamp on the signature, and all files do not have a timestamp (or they have a common timestamp) so there is no risk that the order of scanning can be determined.

15. **Form the Root Hash**
After all the ballots are collected by the scanner, the set of hashes and signatures is placed into an ordered "Hash List"[6]. Then the entire hash list is hashed again, creating the "Top Hash". This block will have the images in known but randomized order, such that it is infeasible to link the ballot to the voter based on the sequence that they were scanned.

[5] The TIFF format is preferred because it is very lightweight and any hash of the complete file is allowed instead of requiring hashes of just the image data, whereas PDF has too much variable overhead that may change even if the image does not change, and so the code images must be hashed and not the complete file.
[6] https://en.wikipedia.org/wiki/Hash_list



A hash list with a top hash

16. **Sign the Root Hash**
    The root hash is signed by the scanner using a stronger signing algorithm such as (for example) RSA-4096. Considering the encoding overhead of JSON and additional metadata, it is reasonable to estimate the size of the JWS (JSON Web Signature) block, including the RSA-4096 public key and the SHA-256 root hash value, to be approximately 600-700 bytes. This final signature will also contain a nonce provided by the EMS when the ballot scanner was deployed.

17. **Cryptographic Data Kept Separately**
    It is important that this block of signed image hashes and the signed root are kept separate from the ballot image data, so it can be communicated and made public prior to the close of polls, because there is no actual voting information exposed in this cryptographic data set. Publishing this data using a transparency service will "lock-in" the ballot images so they cannot be changed. If the data was not published, then there is a small chance that the entire data set could be replaced by a malicious actor.

18. **Low Overhead.**
    To fully secure 1,000 ballot images from a typical precinct, the total space required is 192 bytes per image, plus another 700 bytes for the final signature. For the entire precinct, this is comparable to about the size of one additional image, meaning the total overhead to gain security is about 0.1% of the total size of the unsecured image data. It should be noted that the Concise Binary Object Representation (CBOR) and the related format Concise Object Signing and Encryption format (COSE), if used instead of JWS, reduces the overhead by 30% to 50%. That will be left for further discussion, because although it

is more concise, it is also less transparent and takes one more conversion to use.

19. **Encrypted using Symmetric Keys**
   Per TGP, the data is then encrypted with the stronger and faster encryption algorithm using the randomly generated key originally communicated to the EMS. This process used the EMS's public key, in the first round of communication in the TGP. However, this additional complexity may not be worth it as the public keys could be used for encryption.

20. **Closing the Polls**
   The final step of closing the scanner device is to write a final RATS EAT (entity attestation token) to the flash drive. This will allow the master and auditors to evaluate the final state of the controlled device. Finally, the entire package is signed using the private key of the scanner.

21. **Data Acquired by the EMS**
   Per TGP, the flash devices are then hand-carried from the controlled devices to the TGP master, the EMS. The encrypted data is read by the EMS for further processing. The data is decrypted using the symmetric key provided by the ballot scanner device. This ballot data can be utilized to evaluate the vote on each ballot and create the canvass. TGP includes a full tracking element to make sure all the devices are retrieved.

22. **Immediate Publication**
   As data is retrieved both from deployed scanners in polling places and as ballots are scanned in the central office, cryptographic evidence should be published to a transparency server. This is perhaps only needed on a daily basis and not a batch basis. The cryptographic material is kept separate from the ballot image data for this purpose. A given batch will have:
   a. the signed Hash Root.
   b. The shuffled but ordered list of signed ballot image hashes.
   This publication of in-process checkpoint data can be done to a transparency registry such as that proposed by the IETF SCITT working group. This data should be written to a flash device, also using TGP.

23. **Post-Election Publication**
   After the election has been completed, the complete set of election data should be published. The cryptographic evidence must be logged to a SCITT transparency service or timestamped, using a trusted timestamping service, including:
   a. Public keys list of all ballot scanners used, signed using the private key of the EMS, which must be linked to a trusted root Certificate Authority.
   b. Cryptographic data for each batch:
      i. list of signed hash values of each ballot image (both sides).
      ii. Signed hash of that list.

   c. Cryptographic evidence that all flash devices have been included.

24. **Bulk Election Data**
    Bulk election data can be stored anywhere, but should include a certificate from the transparency service.
    a. Ballot Images in ZIP or equivalent archives, identified so their corresponding cryptographic data can be associated.
    b. CVR - Cast Vote Records of all ballots.
    c. Aggregated totals reports, both in summary and detailed formats.

25. **Audit of the Cryptographic Trail**
    The EMS and other entities can evaluate the cryptographic evidence to evaluate the trustworthiness of the ballot image data, both as the cryptographic data is published and later, when ballot image data is available, as follows for each batch:

    a. The evaluation must utilize the public keys, which must be published earlier in the process.
    b. The returned flash devices must all be accounted for.
    c. The signatures are evaluated and verified of the signed Hash List Root.
    d. The list of signed hashes of each of the images are hashed and compared with the root.
    e. The signatures associated with each of the image hashes is verified.
    f. When the ballot image data is available, it is hashed to verify that it matches the signed hash value for that image.

    Please note that this procedure can detect changes to the image data that occurred at any time after the cryptographic evidence was created, but it cannot correct the data. If any detection occurs, then the paper evidence can be consulted to resolve the discrepancy.

Standards-based approach

A primary goal is to utilize widely accepted public standards, including from the Internet Engineering Task Force (IETF) and documented in standards documents called RFCs "Request for Comments", and from NIST, the National Institute of Standards and Technology, as well as other sources.

https://datatracker.ietf.org/wg/rats/documents/ -- Remote ATtestation ProcedureS (RATS)

https://datatracker.ietf.org/doc/draft-ietf-rats-eat/ -- The Entity Attestation Token (EAT)

https://datatracker.ietf.org/doc/draft-ietf-rats-corim/ -- Concise Reference Integrity Manifest (CORIM)

https://www.w3.org/TR/did-core/ -- Decentralized Identifiers (DIDs) v1.0

https://datatracker.ietf.org/doc/html/rfc7515 -- RFC-7515: JSON Web Signature (JWS)

https://www.rfc-editor.org/rfc/rfc7518 -- RFC-7518: JSON Web Algorithms (JWA)

https://datatracker.ietf.org/doc/html/rfc7519 -- RFC-7519: JSON Web Token (JWT)

https://www.rfc-editor.org/rfc/rfc8152 -- RFC-8152: CBOR Object Signing and Encryption (COSE)
https://www.rfc-editor.org/rfc/rfc7049 -- RFC-7049: Concise Binary Object Representation (CBOR)
https://www.rfc-editor.org/rfc/rfc2986 -- PKCS #10: Certification Request Syntax Specification
https://www.rfc-editor.org/rfc/rfc6234 -- US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF) -- [HMAC = Hashed Message Authentication Codes; HKDF = HMAC-based extract-and-expand Key Derivation Function]
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf -- Security Requirements for Cryptographic Modules
https://www.eac.gov/sites/default/files/TestingCertification/Voluntary_Voting_System_Guidelines_Version_2_0.pdf -- VVSG 2.0 -- Voluntary Voting Systems Guidelines
https://www.iso.org/standard/75804.html -- ISO 23504 PDF/R
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf -- NIST Special Publication 800-63-3 "Digital Identity Guidelines"
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-89.pdf -- NIST Special Publication 800-89 "Recommendation for Obtaining Assurances for Digital Signature Applications"
https://datatracker.ietf.org/doc/html/draft-ietf-teep-architecture-18 -- Trusted Execution Environment Provisioning (TEEP) Architecture
https://datatracker.ietf.org/doc/html/rfc7159 -- RFC-7159: The Java Script Object Notation (JSON) Data Interchange Format

Related Documents
For readers new to the election challenge, the following may be helpful.

https://copswiki.org/Common/M1879 -- "**White Paper: Election Audit Strategy PART 1: Background**" -- describes the current landscape and challenges of our election system in the US. Part 2 evaluates existing Risk Limiting Audit (RLA) algorithm options and suggests the BRAWL method, the Balanced Risk Audit with Workload Limitation. See the document below, which concludes that Ballot Image Audits (BIAs) are the proper target.

https://copswiki.org/Common/M1938 "**The Four Fatal Flaws of RLA Audits**" -- even though RLA audits can work well in some cases, they can't compete with BIAs, except that they do consider the actual paper. RLAs rely on the security of the paper and are limited to at least the same human error rate that will occur in a full hand count, plus the additional risk of sampling. BIAs can include a review of verification scans to provide assurance that the ballot images accurately reproduce the information on the actual ballots.

https://copswiki.org/Common/M1936 -- **Securing Digital Ballot Images to Enable Election Audits (M1936) -** This includes an analysis of hacking scenarios. Most of the concerns here can be addressed by the RATS architecture + SCITT + TGP.

https://docs.google.com/document/d/1Wg1187YW9f_MadLTmspLpikKXOk7TYrzX5d_Ta2Pex4/edit?usp=sharing -- **SCITT: Election Data Use Case** (DRAFT)

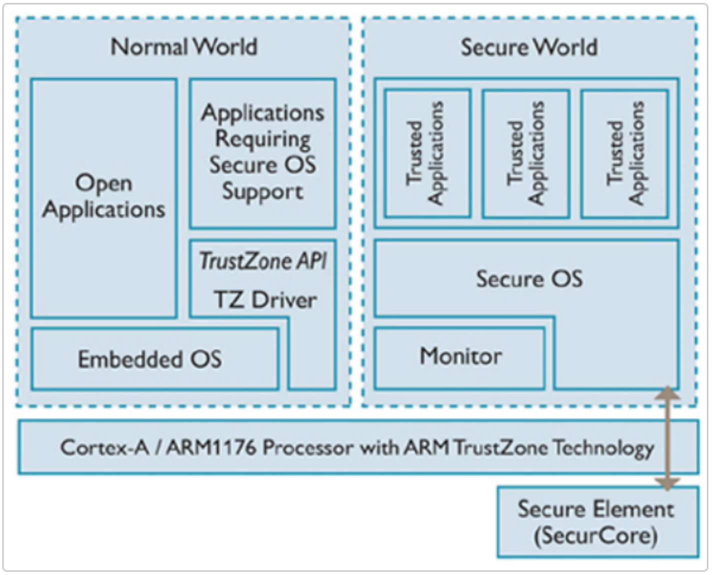
https://docs.google.com/document/d/1xfU_s1Eu51z_WGg5VYBsQtjsKcrV6_TvFXj2WxBcj90/edit?usp=sharing -- **Packaging Arbitrary Data for SCITT.** Analysis of the various options for providing a set of data which includes a number of zip archives that cannot be conveniently all zipped.

For those unfamiliar with Public Key Infrastructure, the following reference is suggested:
Housley, Russ and Polk, Tim "**Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure 1st Edition**" ISBN-10 : 0471397024 (https://smile.amazon.com/dp/0471397024)

Voting Machine Architecture

An important aspect of this proposal is the existence of a trusted core in a given ballot scanner device which has a Trusted Execution Environment (TEE)[7]. The TEE can execute cryptographic and security procedures as defined by NIST FIPS (Federal Information Processing Standard) 140-2 "Security Requirements for Cryptographic Modules", which details how cryptographic systems will operate.

Here is an example of a computer system with a TEE from ARM.



A Trusted Execution Environment (TEE) is a secure area of a computing device that ensures the execution of code can be trusted. The TEE is typically implemented as a separate processor or co-processor, isolated from the main processor; it runs its own operating system and can execute code independently. This separate processor or co-processor is designed to have a higher level of security than the main processor, and is intended to protect sensitive data, cryptographic keys and other assets.

[7] https://en.wikipedia.org/wiki/Trusted_execution_environment

Examples of TEEs include Qualcomm Secure Execution Environment (QSEE), Samsung KNOX and ARM TrustZone. TEEs offer a range of security features such as secure boot, secure key storage, secure communication, and secure execution of code which ensures that the sensitive data remains protected from malicious attacks, malware and other forms of tampering. TEEs are also used to secure the implementation of secure elements, such as SIM cards, embedded Secure Elements (eSE), or Universal Integrated Circuit Card (UICC) in mobile devices.

Although such segmentations can provide a way for the computer to test its own firmware, there is always a set of errors or cases where such self-checking can be thwarted. As usual, security can be improved, but there is no 100% secure system against all attacks.

It is preferable that the ballot scanner include a hardware security module which can generate private keys that are inaccessible by any one or any device. The private key generation occurs internally by random number generation as a “noise source”. This private key is impossible to predict, even by the manufacturer. Support of FIPS 140-2 is mandated by the Voluntary Voting System Guidelines 2.0 (VVSG 2.0):

> 13.3 – Cryptographic algorithms deal with the requirements that cryptographic functionality be implemented in a cryptographic module validated against Federal Information Processing Standard (FIPS) 140 [NIST01]. ...Instead, we propose here that ballot images of hand-marked paper ballots, or ballot summary cards from Ballot Marking Devices (BMDs) are scanned and immediately secured in the scanner. These ballot images are anonymous and yet can be reviewed and retabulated to confirm the results of the election. This approach is an incremental improvement on voting systems that already exist and the security envisioned here can be largely retrofitted into existing equipment.

Further, we propose that any voting machine can be decomposed into a document scanner and possibly a ballot-aware tabulator. Frequently, such a document scanner is embedded in a voting machine, and the machine also includes a ballot aware tabulator that can recognize the darkened ovals on the ballot. The physical scanner element is likely a subassembly that is provided by a document scanner manufacturer to the voting machine manufacturer.

It is our position that improved security can be obtained if the document scanner is either not combined with a ballot-aware tabulator, or the scanner should be internally separated from the ballot-aware tabulator. Thus, it would be much more difficult to include a hack that would alter or substitute images, because the scanner element is simply not aware of what documents are being scanned, nor what one darkened spot might mean over another. The scanner can make high fidelity document images but does not understand what is on them. This is analogous to an eagle with very sharp eyes and yet cannot read. No matter how well the eagle can see, it doesn’t have the processing capability to read. A high-fidelity scanner that can create images and secure them without having the capability to read ballots will improve security. Such systems can be more easily proved to be secure.

Today, it is a common configuration that commercial off-the-shelf (COTS) scanners are used in central scanning operations and sometimes in voting centers or precincts. However, these scanners do not typically provide the high security envisioned here and sometimes have potential security leaks due to the use of drivers like TWAIN, WIA, ISIS, etc. which may provide hacking opportunities. We suggest that the scanner device include a "Trusted Execution Environment" which is associated with the FIPS 140-2 compliant cryptographic security module.

In the description below, we will also refer to the **Election Management System (EMS)** which is a software application or platform used to manage and administer various aspects of an election process, including voter registration, voter identification, ballot design, vote counting, and results reporting. In conjunction with this central tabulator function, there are also "voting machine" scanners which are deployed to polling places throughout the jurisdiction.

A **certification authority (CA)** is an entity that issues digital certificates, which are used to establish a secure connection between two endpoints, such as a website and a user's browser. These digital certificates are used to authenticate the identity of an endpoint and to (possibly) encrypt the data that is sent between the endpoints. A CA is a trusted third party that verifies the identity of the endpoint and issues the digital certificate. Some well-known Certification Authorities include DigiCert, GlobalSign, and VeriSign.

A **subordinate CA** is responsible for issuing digital certificates to endpoints or devices, but it does not have the same level of trust as the root CA. It is "subordinate" to the root CA in that it relies on the root CA's digital certificate to establish its own trustworthiness. The root CA issues a certificate to the subordinate CA, which is then used to sign the digital certificates that the subordinate CA issues to end-users or devices. In this case, we are envisioning that an EMS would operate a subordinate CA for the purpose of tracking the public keys of deployed voting machine scanners. However, there may be other alternatives for maintaining the public keys that correspond to each ballot scanner and therefore its private key.

With that as background, we now describe how such a system can scan the ballots, create ballot images, and ensure that the ballot images cannot be modified without detection after they are properly cryptographically secured.

Conclusion

The presentation of this methodology for improving election systems cybersecurity provides the general steps required for a complete system. Along the path to progress, this will include further refinement and consensus-based methodology to incorporate these improvements as soon as possible.

APPENDIX 1: TransGapProtocol

The TransGapProtocol (TGP) document was not ready for release for the VVSG comment period. However, a conceptual description is provided here to provide to the reader a better idea of the goals and requirements.

Goals and Characteristics of the TGP:

1. Provide a standard methodology for using removable media, typically flash memory "thumb drives" or "jump drives" to securely track data provided to, from, and between air-gapped computer systems.

2. Internal to one removable media can be viewed as a file system. In that file system can be a "security_log" folder, and a "content" folder. At the highest level, these folders can be zipped, but in many cases, the content is already zipped and may be quite large in size. Thus, additional zipping is not recommended.

3. The content folder can include any content using a file system layout that is convenient for the application.

4. The security log folder provides an implementation of an append-only log with security information which is hash-linked together.

5. For the election systems application, the security log will contain the following:
    a. Public key of the master system
    b. Nonce random number
    c. signature of the above.
    ------- device is plugged into controlled system
    d. ID of the controlled system
    e. public key of the controlled system.
    f. symmetric encryption key encrypted by EMS public key.
    g. RATS EAT block providing verification data from the controlled device.
    h. response Nonce.
    i. signature by the controlled device seeded by Nonce of (b).
    ------- device is plugged into master system
    ------- public keys are read from controlled devices and published.

    j. data request from the master system.
    k. Nonce random number
    l. configuration data
    m. signature by master system
    ------- device is plugged into the controlled system
    n. ID of the controlled system

o. Initial RATS EAT block by controlled system providing verification data.
p. root hash of the hash-list of all images scanned.
q. Final RATS EAT block by controlled system providing verification data.
r. signature by controlled system
   ------ device is plugged into the master system
s. final closing record added to the device.
t. master system signs everything.

6. The security log, in its entirety, for each controlled device is an artifact that is published to a transparency service.

APPENDIX 2: Notes for Development Reference Only

The following additional details are provided but many of these ideas are instead accomplished by the general TransGapProtocol (TGP).

Prior to Deployment

1. The air gapped (not connected to the internet or any outside computers, and has no wireless capability) EMS shall maintain a database of public keys used by the scanners used in the election. There are several options for performing this:
    a. operate a **subordinate CA** for the purposes of managing the signed data being returned. Since the EMS operates in an air-gapped environment, it is not possible to contact a centralized CA during the election. However, prior to the election, the EMS can be authorized by a parent CA such as GlobalSign or other organization that offers such services, such that it can operate a subordinate CA in the air-gapped network in the jurisdiction. The functionality of a subordinate CA includes the function that private keys are associated with specific devices.

    b. using Decentralized Identifiers (DIDs)[8] methodology with SCITT transparency service registry. There may be some issues with this approach that still need to be resolved.

       Decentralized identifiers (DIDs) are a new type of identifier that enables verifiable, decentralized digital identity. A DID refers to any subject (e.g., a person, organization, thing, data model, abstract entity, etc.) as determined by the controller of the DID. In contrast to typical, federated identifiers, DIDs have been designed so that they may be decoupled from centralized registries, identity providers, and certificate authorities. Specifically, while other parties might be used to help enable the discovery of information related to a DID, the design enables the controller of a DID to prove control over it without requiring permission from any other party. DIDs are URIs that associate a DID subject with a DID document allowing trustable interactions associated with that subject.

    c. Using the Remote ATtestation ProcedureS (RATS). This protocol appears to have at least some of the required building blocks, but does not appear to handle the key management aspect.

These options are still being studied and not all the issues are fully resolved. Below, for the purposes of setting forth the idea, the method of (c) and (a) are described.

2. When a new scanner device is provisioned for use by a given EMS, while in a secure configuration, most likely at the election office, it is initialized using the flash memory device and air-gapped from any public network. This configuration follows the "passport"

[8] https://www.w3.org/TR/did-core/ -- Decentralized Identifiers (DIDs) v1.0

RATS operational model.

3. The scanner device will format and initialize the flash memory device, and it will provide an Entity Attestation Token (EAT) message with information about the device and provide its private key. Also included in the flash drive is a requestAttestation message per IETF RATS (Remote ATtestation ProcedureS specification).[9]

   IETF RATS (RFC 9334) is a protocol for Remote Attestation of Things, which provides a standard way for devices to prove their identity and the authenticity of their software to a remote attester. The requestAttestation message is a specific element of the RATS protocol that is used to initiate the attestation process. (This comment may be related to an older version of the spec.)

   The requestAttestation message contains information about the device that is being attested, including its identity, the software that is running on it, and the level of assurance that is required for the attestation. It also includes a timestamp and a nonce, which are used to ensure that the attestation is current and that it has not been tampered with.

   Once the requestAttestation block is received by the remote attester (the EMS), it can verify the information contained in the block and, if it is satisfied that the device and its software are genuine, it can issue an attestation certificate. This certificate can then be used to authenticate the device and the software to other parties.

   It is important to note that the requestAttestation block is just one of the components of the RATS protocol, which also includes the Attestation Result and the Attestation Evidence.

4. The EMS reviews the EAT from the scanner device and writes a RATS certificate to the flash device, along with configuration data for this given election.

5. The EMS adds to the flash drive configuration parameters for the election. A given flash drive is paired with a given voting machine scanner. In option (c) also included here is the RATS certificate that provides sufficient information for future data retrieval.

6. In option (a), it also contains values appropriate for the Certificate Signing Request (CSR), such as the

a.	organizationName	"Registrar of Voters"	
b.	jurisdiction	"XYZ County"	
c.	stateOrProvinceName	"ST"	(state abbreviation)
d.	streetAddress	"123 Main St."	
e.	localityName	"City of TUV"	

[9] https://datatracker.ietf.org/wg/rats/documents/ -- Remote ATtestation ProcedureS (rats)

f.	postalCode	99999-9999
g.	telephoneNumber	123-456-7890
h.	nonce	948579387593847598304985

The following is hardcoded in the machine:

i.	serialNumber	34859843765983457 (hardcoded in the machine)
j.	manufacturer	ES&S (for example)
k.	model	DS200 (for example)

The flash drive is mated to one machine and cannot be used in any other machine. This is similar to the data provided to scanners in a centralized scanning operation.

7. RATS EAT (Entity Attestation Token) Components are initialized by the scanner device and written to the flash memory device, as follows:
    a. **eat_nonce (EAT Nonce) Claim** -- minimum 64 bits
    b. **ueid (Universal Entity ID) Claim** -- The "ueid" claim conveys a UEID, which identifies an individual manufactured entity, in this case the scanner device. UEIDs MUST be universally and globally unique across manufacturers and countries. UEIDs MUST also be unique across protocols and systems, as tokens are intended to be embedded in many different protocols and systems. No two products anywhere, even in completely different industries made by two different manufacturers in two different countries should have the same UEID. The UEID is permanent. It MUST never change for a given entity. A UEID is up to 33 bytes.
    c. **oemid (Hardware OEM Identification) Claim** -- identifies the Original Equipment Manufacturer (OEM) of the hardware. There are three types:
        i. Random number based (16 bytes)
        ii. IEEE Based OEMID (3 bytes)
        iii. IANA Private Enterprise Number Based OEMID (3 bytes)
    d. **hwmodel (Hardware Model) Claim** -- (1 to 32 bytes) The "hwmodel" claim is for use in protocols and not for human consumption. The format and encoding of this claim should not be human-readable to discourage use other than in protocols. If this claim is to be derived from an already-in-use human-readable identifier, it can be run through a hash function.
    e. **hwversion (Hardware Version) Claim** -- text string the format of which is set by each manufacturer. The structure and sorting order of this text string can be specified using the version-scheme item from CoSWID. It is useful to know how to sort versions so the newer can be distinguished from the older.
    f. **swname (Software Name) Claim** -- The "swname" claim contains a very simple free-form text value for naming the software used by the entity. Intentionally, no general rules or structure are set. This will make it unsuitable for use cases that require precise naming.
    g. **oemboot (OEM Authorized Boot) Claim** -- An "oemboot" claim with value of True indicates the entity booted with software authorized by the manufacturer of the entity as indicated by the "oemid" claim. This must be True in this application.

h. **public_key Claim --** This claim is not enumerated by the RATS architecture documents, but there is a requirement, in the RATS architecture, that the attesting device provide the public key that corresponds to a private key, that is not available to anyone, in the device receiving attestation. The private key may be generated using a Hardware Security Module (HSM) compliant with FIPS 140-2. Note: if the public key cannot be communicated in this manner, then the CSR method of X.509 can be used. An explanation of this method is provided in section 10, just below.

   The current plan is to use TGP to communicate public keys and optionally use RATS for device-level information.

8. The scanners are then deployed to the polling locations in the district.

At the deployed location:

9. The flash device, now with RATS Certificate, is plugged into the machine and it is powered on. The RATS Certificate and configuration information is read from the flash memory device. The voting scanner will create an Entity Attestation Token (EAT)[10] relevant its initial state and write this to the flash drive.

10. If it is not possible to use the RATS protocol to establish the public key, the X.509 method can be used, as follows:

    The voting machine will create a Certificate Signing Request (CSR) PKCS#10[11] PKCS#10 CSR is a standard format for requesting X.509 certificates from the Certification Authority. The machine will include the fields provided in the configuration flash drive and will also provide the hard-coded serial number, mfr, and model of the device. It will sign the CSR using the private key.

    The voting machine will write the CSR to the flash drive.

11. For each ballot scanned, the scanner should save the image without extensive image processing. Limited image recognition to determine image quality and front/back orientation and rotation is allowed. Since a ballot sheet may be scanned in any orientation (top/bottom first, front/back up), the images may be rotated and the sides placed in order, however, this is optional. Additional processing of the images to recognize the vote is optional. The ballot images can be of hand-marked paper ballots or ballot summary cards from a Ballot Marking Device (BMD).

[10] https://datatracker.ietf.org/doc/draft-ietf-rats-eat/ -- The Entity Attestation Token (EAT) -- draft-ietf-rats-eat-17
[11] https://www.rfc-editor.org/rfc/rfc2986 -- PKCS #10: Certification Request Syntax Specification

12. The images saved shall not have additional processing to align or register the images. However, the voting system scanner may do other processing to prepare the image for mark recognition, if required, but that processing shall not modify the original image.

13. The image data should be prepared to be identical to the data when wrapped in PDF or TIFF file wrappers, as used by the specific voting system. Images in TIFF files used by Dominion[12] and PDF files used by ES&S[13] both use CCITT Group3/4 lossless compression as described by ITU recommendation T.4[14]. Lossy compression (such as JPEG) or JBIG compression shall not be used. Non-bilevel, single-bit-per-pixel images would require additional scrutiny before allowing them to make sure that no data can be hidden in the image. The PDF/R (raster) format may be a good match for this use case.

14. Although it is not required that the ballot image data use a PDF or TIFF wrapper in the flash drive, placing the data in that format at this stage is allowed. Please note, TIFF is a deterministic format, in that it will have the same hash value for the same image content and metadata. PDF is not, however. No metadata is allowed other than the image size, bits per pixel and other parameters. No timestamp is allowed, or if provided, it must be set to a constant value for the election. Voting systems deployed to the field are required to shuffle the images and assign numbers that will not allow the sequence of images to be paired up with the order of voters, if that order is known.

15. The scanner shall then create either a) a single hash value for the image data of the two sides which are combined into one logical image (such as by calculating the hash of page 1 and then continuing to calculate the hash for page 2, or other method) or b) two hash values, one for each side. The hash algorithm should be SHA256[15]. The format of the image data hashed must be the same as the data ultimately wrapped in the PDF or TIFF wrappers for the two sides of the sheet. Any additional data, such as the third-page "AuditMark(tm)" as used by Dominion shall not be included in creating the content digest hash value. The hashed data also does not include the PDF or TIFF wrapper, unless these can be sufficiently standardized and do not include any mutable data.

16. **ImageCOSE files**: The content digest hash value(s) shall be signed using the scanner's private key to create the signature token. The COSE (CBOR Object Signing and Encryption)[16] data format shall be used to sign a CBOR (Concise Binary Object Representation)[17] object that contains the hash value(s) of the image data. The CBOR object shall not also contain the image data.

[12] Dominion Voting Systems -- https://www.dominionvoting.com/
[13] Election Systems & Software "ES&S" -- https://www.essvote.com/
[14] https://www.itu.int/rec/T-REC-T.4-200307-I/en -- Standardization of Group 3 facsimile terminals for document transmission
[15] https://www.rfc-editor.org/rfc/rfc6234 -- US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)
[16] https://www.rfc-editor.org/rfc/rfc8152.html -- CBOR Object Signing and Encryption (COSE)
[17] https://www.rfc-editor.org/rfc/rfc7049 -- Concise Binary Object Representation (CBOR)

17. The voting machine will also encapsulate and sign the aggregated results, if they are derived from the image data.

18. The voting system then begins closing procedures.

19. Data is finalized and any final data is written to the flash drive.

20. The voting machine will create an Entity Attestation Token (EAT)[18] relevant its final state and write this to the flash drive. This attestation includes the use of the nonce provided.

Data is returned to the EMS

21. The data from each voting machine scanner is returned to the Election Management System (EMS) in the central office by hand-carrying the flash device. Optionally, the signed aggregated results can be returned by cell modem transmission, if such transmission is allowed by state law. Such transmission is not recommended, however.

Central Scanning is Similar

22. For centralized scanning operations operating on an air-gapped LAN (local area network), each scanner must use the RATS protocol to communicate its public key and initial state (or prepare an X.509 CSR) then write an EAT related to its initial state, and communicate it to the EMS over the LAN. The operation is essentially the same but the interaction with the local CA can be more robust, and may include more rounds of handshaking, because it is connected to the EMS using the air-gapped LAN.

23. For centralized scanning operations, the scanner must create a signed hash value for each image as it is produced and within the scanner itself. The signing of the image hash must be performed within the security of the scanner and not after it is transmitted via a scanner interface, such as ISIS or TWAIN, rather than by a device driver in the host.

24. For centralized scanning operations, the scanner must create an EAT for each batch processed, providing attestation of current state at that time.

25. As the election proceeds, the election office can publish the imageCOSE files regarding the ballot image data and the EAT files without providing the image data. This is particularly important when scanning ballots received early. Publishing the imageCOSE files will secure the images without providing the image data. By publishing as they go, there isn't any opportunity to alter the data.

26. For each thumbdrive returned or scanner used in a central scanning operation, the EMS shall validate the RATS Certificate (or process the CSR using the public key included in the CSR block, including verification of the randomly selected nonce for this election and for this machine). It will also verify the initial and final EATs. If using the X.509 option,

[18] https://datatracker.ietf.org/doc/draft-ietf-rats-eat/ -- The Entity Attestation Token (EAT)

after verification, it will submit the CSR to the local subordinate CA within the air-gapped network to create a repository of signed certificates. Each certificate also specifies the specific machine that has the matching private key for the public key in the certificate. Eventually, this information is made public.

27. The EMS will accept the image data, and optionally the aggregated results and cast vote records (CVR), if they are produced by any ballot-aware scanner devices, all of which are signed using the private key in the machine. It verifies that this machine created the image and results data by calculating the hash value over the data, and verifying the signed hash value by using the public key. The public keys of each device are published as well.

28. The EMS may perform additional organization, shuffling, and may renumber all the records. The images can then be placed in TIFF or PDF wrappers, if they are not already used. If TIFF is used, it will use a single "strip" containing all the rows for each image, and not use the option of breaking the image into multiple strips. The image data will remain in the same CCITT compression as was originally provided. (TIFF does not provide any internal fields for signing the file.) PDF may optionally use the internal security mechanisms, but it should be over only the image data and not other aspects of the wrapper, so the signed hash values can be the same as the imageCOSE values.

29. For each image file, an imageCOSE file will be created which provides the signing parameters and the content hash values. The imageCOSE file does not contain image data.
    The signed CBOR block contains:
    a. The byte offsets within the (PDF or TIFF) file to the image data (List of integer pairs, specifying offset and length of each segment to be included).
    b. The content hash(es) of the image data specified above.
    c. The signature.

After the Election:

30. Once all the ballots have been cast, then the ballot images are combined into ZIP archives with a target of about 5 to 10 GB per file, up to about 50,000 ballot images per file. Also included are the imageCOSE files which have similar names as each image file. These imageCOSE files may be in a separate ZIP file so they can be published prior to publishing the images.

31. All ballot image ZIP files, Cast Vote Records (CVR), other results, the certificates, and EATS for each scanner are provided on the jurisdiction website or other posting service. There is a manifest file which contains the file names, and SHA512 hash for each ZIP and CVR file as a whole. This issue is separately investigated in this document: "Packaging Data for SCITT". It is important that these files are not too large because

calculating the hash values can be difficult otherwise.

32. The summary of the packaged data will be the item submitted to an immutable ledger SCITT, but this is still under study.

33. An auditing service can then check on the integrity of each image file by comparing the image data with the hash values in the imageCOSE block, and it can use the public keys, which are also published, to verify that the machine signed each image with its private key.

Requirements for SCITT:

Here is a list of requirements to keep in mind regarding the requirements of this use-case in SCITT, the Supply Chain Integrity, Transparency and Trust working group.

.

1. Reference to data which exists outside the registry must be possible because of the size of the data (between 5GB to perhaps a few TB). However, if the SCITT ledger is paired with storage that is sufficient, it could be used. Today, typical jurisdictions that do post data post aggregated results to their own web server and the underlying data is posted to a posting service that can handle larger files, such as sync.com, dropbox, etc.

2. Air gapped operation using TGP. During the initialization and operation of the voting systems, they are operated without any connectivity to the Internet, with initialization and resulting data transferred using TGP. Eventually, the data from the election will be placed on a posting service that can be accessed by the public.

3. Time phased operation, where early and incomplete results can be posted and then complete results can be posted later, including legal certification of the final results by various entities. This could be done with two or more SCITT submissions, and implies that perhaps the option of operating a SCITT ledger by the EMS may be attractive.