End-to-End Verifiable Internet Voting Blockchain

Concepts: A proposed specification and feasibility assessment study with respect to Absentee / Remote Voting by the US Military

Bill Edwards
Deep Blue Krypto Group
June, 2018

Abstract. A certain disruptive technology called blockchain can be a transformative technology for various aspects of the US Government. A blockchain, which can store data as well as transactions — meaning clients no longer need to trust third parties with their data. An information age platform for the future — providing much-needed stability, scalability, security and functionality for the Federal space. Efficacies of elections are improved as election integrity can be ensured, confidentiality of voters' choices can be ensured, and the validity of the results can be safeguarded. Every eligible individual should be able to actively participate in democracy by easily and safely voting when, how, and where they want. Democratic institutions generally subscribe to this philosophy; yet do little to implement it in practice. Based on new technologies, you can request an absentee ballot, and in some jurisdictions actually complete it, right from your own mobile phone or secure web browser. Currently, most blockchain platforms are used in the financial applications,

however more and more new applications for different fields start to appear. Applications that require high reliability and full elimination of data manipulation risks can use blockchain. In addition, blockchain is distributed, so it avoids the single point-of-failure situations. Consequently, the challenges faced by military voters are immense. As America's most mobile population, military voters are constantly on the go moving from one duty station to the next. If they have any hope of voting, military voters are required to navigate a confusing array of state absentee voting laws. In many cases, the request for an absentee ballot never comes or comes too late to vote. We propose an evaluation and experimental study of a PKI blockchain voting system which could be recreated in context and verified indefinitely after the end of the election by validating the results and all signed vote transactions, which will suggest and confirm a proposed framework that provides more reliable and robust blockchain PKI systems for the US Military regarding electoral security.

Introduction. Obstacles to Internet voting range from the risk of hacking to political and policy considerations. Scientists, federal agencies, cybersecurity specialists, and certain organized activists in the U.S. have urged against exposing the ballots of the most powerful nation on Earth to the seemingly endless range of threats that lurk on today's Internet. Election integrity advocates say that secure, tested, certified remote voting systems are not available. Cybersecurity specialists do not consider online ballot return systems secure, and no Internet voting systems have been certified at the time of this writing. Many election administrators have turned to email as a method for ballot transmission, although it does not provide the benefits that a secure, full-featured voting system would. Email is not secure, yet election administrators and voters regularly use it to transmit ballots because no viable alternatives are available.

Many people believe that Internet voting will increase voter participation, help with voter decision-making and engagement, provide equal opportunity for voters with disabilities, and decrease election costs. Trustworthy democracy is a worthwhile goal, and we should strive to achieve it. The only responsible way to make progress is to continue peer-reviewed research and experimentation. The development and deployment of a high-assurance distributed system of the scale and import of a public end-to-end verifiable election system has never been attempted, especially using a secure blockchain.

Blockchain has the ability to store and manage digital certificates within a public and immutable ledger, resulting in a fully traceable history log. In this proposal we investigate multiple designs and developments of blockchain-based PKI management frameworks for issuing, validating and revoking X.509 certificates for Secure Online End-to-End Verifiable Blockchain Voting. Evaluation and experimental results would confirm that the proposed framework provides more reliable and robust PKI blockchain voting system could be recreated in context and verified indefinitely after the end of the election by validating the results and all signed vote transactions possible using a secure mobile device. Furthermore, blockchain end-to-end verifiable voting means that every action that transpires, verified in real-time by the entirety of the network, is inviolable once the transaction that represents that action is written to the blockchain. Every action by a voter is fully visible to the voter and to every node at any time, maximizing visibility into an ongoing election without sacrificing the voter's anonymity. The project will show the means by which the new platform allows voters with accessibility considerations to use their mobile devices to independently receive election alerts, research candidates and issues, access their correct ballot styles, listen to the instructions and choices read aloud, mark their ballots on screen, sigh their completed ballot packages onscreen, and return their ballots electronically from their private email accounts to their election administrators, all without the assistance of other people.

The project will focus on fully compliant policies such that thoughtful integration of mandates of the federal and recognized access standards including the ADA, HAVA, WCAG 2.0, the Rehabilitation Act and others. Every eligible individual should be able to actively participate in democracy by easily and safely voting when, how, and where they want. Current democratic institutions (local, state and federal) generally subscribe to this philosophy; yet do little to implement it in practice. An end-to-end verifiable voting system allows voters to: 1) check that the system recorded their votes correctly, 2) check that the system included their votes in the final tally, 3) and count the recorded votes and double-check the announced outcome of the election. It is not enough for election results to be correct. To be worthy of public trust, an election process must give voters and observers compelling evidence that allows them to check for themselves that the election result is correct and the election was conducted properly. Open public review of the entire election system and its operation, including all documentation, source code, and system logs, is a critical part of that evidence.

Internet voting systems currently exist, but independent auditing has shown that these systems do not have the level of security and transparency needed for mainstream elections. Security experts advise that end-to-end verifiability—lacking in current systems—is one of the critical features needed to guarantee the integrity, openness, and transparency of election systems.

In this study project, we examine the future of voting and the possibility of conducting secure elections online.

Objectives. The objective of this technology/protocol is to help election management bodies introduce online voting with levels of security and safety that are superior to in-person, paper-based elections, with restored trust, increased access, voter verifiability, and better public transparency. A case in point is for DHS and the DoD ensures that challenges faced by America's most mobile population, military voters are constantly on the go moving from one duty station to the next. If they have any hope of voting, military voters are required to navigate a confusing array of state absentee voting laws. While the military abroad have been subject to an incredibly logistically complicated process, with limited transparency, reliability, and access, that does not have to be the case. Therefore this platform should be compatible with any iOS and Android mobile device, easy-to-use absentee balloting app is the easiest and safest way to cast your absentee or remote ballot.

By applying the pertinent properties of blockchains for online voting, the appropriate secure voting platform would cryptographically secure a protocol that is able to guarantee:

- Independently Verified Results: In traditional paper ballot voting, the voter must trust the poll workers and local election officials to ensure that their vote is recorded. Blockchain networks allow for multiple nodes to independently ensure that the vote is recorded.

- Vote Accuracy Immune from Compromise: In traditional paper ballot voting, the voter must trust the poll workers, and local election officials to ensure that their vote is not tampered with. Blockchain networks enforce immutability through the cryptographic integrity of Merkle trees and chained references.

- Public End-To-End Verifiability: In traditional paper ballot voting, the voter has no means to verify that their vote is counted in the final tally. Blockchain allows voters, as well as independent auditors, to independently verify that their vote has been collected and will be included in the final tally.

In the recent years blockchains have been evolved to allow the execution of arbitrary logic known as Smart Contracts. Conceptually the smart contract is an application that runs on the top of blockchain and uses the underlying ordering of transactions to keep consistency of smart contract execution results between peers. For example, Ethereum blockchain supports complex and stateful Turing-complete language, like Solidity1, that can be used to program and define wide range of application scenarios.

The link between a given certificate and the Root Certificate is known as a Chain of Trust. Importantly, chain of trust may include any number of sub-CA certificate between a given certificate and the Root-CA certificate. However, the X.509v3 has an extension named Basic Constraints and this extension can limit the maximum depth of valid certificate chain (chain of trust). Blockchain-based PKI framework supports the revocation of the certificate, which is a real issue in the traditional PKI systems. Moreover, as it is impossible to delete information from blockchain, only a parent CA can mark a certificate issued by him as revoked. Thus, any misbehavior of a CA regarding certificate revocation will be also traced and noticed by all other entities.

Secure online voting for overseas citizens (military) eliminates the risk of:

- lost or damaged mail, mail delays and disruptions such as postal service strikes
- ballot cards sent to the wrong address or discarded because they were filled out incorrectly or ambiguously
- hanging chads
- fraudulent votes privacy risks.

Law. In the U.S., legislators must change the legal framework of elections in nearly every jurisdiction that wishes to use Internet voting. Historically, legislatures are comfortable with introducing Internet voting trials, particularly for UOCAVA voters. Providing technology that helps disabled voters is commonplace.

However, legislatures often permit or mandate the use of new election technologies with little restriction on their form, substance, and impact. This creates serious problems. Legalizing Internet voting without mandating end-to-end verifiability, or permitting large-scale Internet voting without first evaluating the impact and success of this concept in polling places and in small-scale Internet voting trials, could have disastrous consequences.

Based upon historical evidence, a gradual evolution of state and local election law—particularly facilitated by the local nature of elections—seems feasible in a 5–10 year time frame.

A subsequent research and development phase of this project should include concrete legal recommendations for state and local legislators. These recommendations must ensure that the legal framework for Internet voting deployment is rational, evidence-based, and legally mandates the requirements set forth in this report**.**

Politics. In the main, politicians and elected officials want to be perceived as forward thinking and modern. Thus, it is not uncommon for those running for office to support new election technologies such as Internet voting. On the other hand, political parties and powerful political special interest groups are motivated by other factors. The hypothetical implications of widespread trustworthy use of new electronic blockchain mobile application—particularly the possibility of increased broad-spectrum voter participation—are potentially at odds with the agendas of some political actors.

The Trust Model. Blockchain applications specifically address one of the most difficult factors challenging electoral integrity – the trust model. Blockchain ensures trust is distributed amongst a set of mutually distrustful parties, all of who are potentially adversarial, that participate in jointly managing and maintaining the cryptographically secure digital trail of the election. By distributing trust in this way, blockchains create a trustless environment whereby the amount of trust required from those participating in an election is minimized.

One characteristic of the blockchain inertia is to bring online voting to the mainstream, especially for absentee voting and military oversees voting procedures. The technology behind elections is hard to get right. Elections require security. They also require transparency: anyone should be able to observe enough of the election process, from distribution of ballots, to the counting and canvassing of votes, to verify that the reported winners really won. But if people vote on mobile devices or votes that are tallied by computers, key steps of the election are not transparent and additional steps are needed to establish fair and transparent elections around the world with end-to-end verifiable blockchain voting technology.

Blockchain voting allows for every participant in the network to verify every transaction written to the digital audit trail. This is advantageous in online voting systems as it means end-to-end verification of the election happens on a rolling, ongoing basis during the election process itself. While the use of blockchain in voting allows for ongoing verification of an in-process election, it also allows for a retrospective end-to-end verification of all election activities after the election has finished. Elections administered using a blockchain voting system could be recreated in context and verified indefinitely after the end of the election by validating the results and all signed vote transactions.

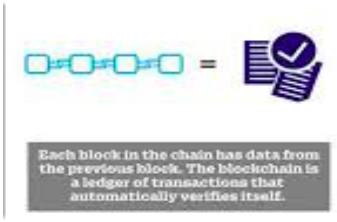
The question we should be asking is "how can we ensure that election results are accurate when we cannot trust the computers used to run elections?" rather than "how do we make electronic voting secure?" Nothing is ever absolutely secure. But we can often make computers, systems, and processes "secure enough" for their tasks, provided we have an independent way to check the results. One technique is to produce a voter-verified paper trail, ensure that the paper trail is trustworthy, and manually audit the electronically tabulated results against the paper audit trail. Another technique called "end-to-end verifiability" allows individual voters to verify that their vote was recorded and counted correctly. Simply enabling everyone to examine the source is not sufficient, and could even give voters and election officials a belief that the system is secure when it is not.

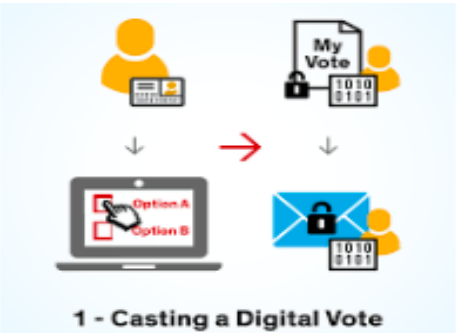
The project proposes and asserts that we need a blockchain voting solution that is architected to meet the performance needs of a mission critical election. We propose that this disruptive technology strives to meet the evolving needs of modern voters, especially in the military. We suggest that over the long run, we

seek to enable any authorized voter to participate in an election through their own digital device, all while guaranteeing the security and transparency of the electoral procedure. To understand how our technique to blockchain voting succeeds where traditional systems have struggled or even failed, we have developed a template of characteristics that are necessary for election results to be trusted. A free and fair election must minimally satisfy the following requirements: transparency, privacy, integrity, affordability, trust, and accessibility.

Furthermore, blockchain voting means that every action that transpires, verified in real-time by the entirety of the network, is inviolable once the transaction that represents that action is written to the blockchain. Every action by a voter is fully visible to the voter and to every node at any time, maximizing visibility into an ongoing election without sacrificing the voter's anonymity.

The mission. A voting platform solution needs to satisfy all of the requirements that we believe are necessary to ensure a free and fair election, including transparency, privacy, integrity, trust, affordability and accessibility. An open-source protocol that lays the technological foundation for verifiable elections, accessible elections, secure election, and transparent elections is what needs to be studied as well.





Blockchain is the key technology that unlocks this mission. Blockchain provides a trustless, digital and decentralized method of generating cryptographically secure records, which also preserve the anonymity of participants while remaining open to public inspection. Applied to voting, blockchain ensures that votes are recorded accurately, transparently, permanently and securely.
It can be challenging to implement complex voting technology that serves an entire nation's population. Furthermore, with a diverse array of laws, election rules and voting frameworks between local, state and federal,

For example, millions of citizens are living, traveling, studying, working and serving overseas (classified under the Uniformed and Overseas Citizens Absentee Voting Act – UOCAVA Citizens). Their votes can be crucial in deciding an election outcome in their home country. This might be a solid use-case for this study.

The study will ascertain if the use of a custom solution offers a solution based on salient requirements or one that has been developed with an open-source core platform that will offer each voting administrator the ability to integrate this technology into its own electoral procedures. This study should establish a solution that can enable secure and remote voting from digital devices, including personal computers and mobile phones. The ultimate goal is for voters (our use case could be the US Military) to be able to vote from anywhere using this technology, removing the need to physically travel to polling stations in order to participate in an election. A mobile app solution such as this better fits the lifestyle of modern mobile voters, who are presently required to use outdated voting techniques. Therefore, the importance of accessibility goes beyond simple convenience and creates new ways of ensuring election fairness. There have been numerous recorded incidents within the US in which valid voters have been prevented from participating in an election because of the actions of identity, fraud and third party intervention. The ability to vote from a personal device outside of an election facility can mitigate the impact these groups may have on an election. The central strength of any blockchain solution is cryptographic security. Maintaining the integrity of the elections that occur on any network or the Internet is of the utmost importance to this study, and the selection of this coordinated effort to find the appropriate technology(s) has been built to transparently ensure this. As a result, any third party at any point throughout the voting process cannot alter ballots and final election results. Blockchain, the technology we choose to study, is the key component of a solution architecture that protects against intervention from the local, state and federal government, institutions, third parties and others who may seek to subvert the election process.

Technology. This study will assess the various disruptive technologies that form a mobile voting platform. The study should focus on a multi-layer architecture that is based on blockchain technology, which includes several innovations that have been developed by open-sourced vendors that provide a high level of immutability and decentralization of data. The system/platform should be architected which will provide high throughput capabilities and low overhead, which enables the platform to be run on low bandwidth devices. It is our study assumption that these layers communicate with each other at various instances throughout the election process to provide a cryptographically secure voting environment with auditable proofs. This blockchain network provides an immutable record of all data throughout the election process and acts as the central communication channel, memory and permanent data store of the system. An idea comes to mind when studying the technology behind blockchains. For instance, the blockchain itself (is the permissioned blockchain of the MilDep/DHS networks, which consists of write-permissioned nodes operated by the MilDep/DHS and recognized third-party witnesses (Consensus Nodes) as well as read-only nodes (Citizen Auditor Nodes) that can be operated by anyone in the world. This blockchain network provides an immutable record of

all data throughout the election process and acts as the central communication channel, memory and permanent data store of the system.). Every election system must guarantee the privacy of its voters. For this study in particular, the system must ensure that votes cannot be linked to individual voters when votes are decrypted for tallying. In that respect, once the voting period ends, the network runs all ballots through a mixing network to anonymize the encrypted ballots cast on the permissioned blockchain.

Making a case for PKI. The best solution for protecting encryption from our ever-growing processing power is integrating decentralization into Public Key Infrastructure. The purpose of this study is to analyze several important use cases based on blockchain, including: blockchain based PKI for security connection, blockchain as a service, interworking cross blockchain (exchange data and contracts cross chains). Through use case analysis, important scenarios and specific requirements will be discussed. Related solutions are also provided for easy understanding.

The value of blockchain in building a system of trust and collaboration has been proved by the industry, and more and more enterprises and industries are applying the blockchain. However, not all enterprises are willing to establish and operate their own blockchain system. Therefore, providing blockchain as a service contributes to the rapid popularization of blockchain, especially for this study. The study proposes a blockchain-as-a-service model.

The main requirements for blockchain-as-a-service may include:

- General proposed data format to support different applications, for storage/verification/resolve
- Define flexible, extensible interfaces/APIs that is easy for programming
- Enhanced performance for future proof

It is significant to make a case for using current IETF specification profiles the format and semantics of certificates and certificate revocation lists (CRLs) for the Internet PKI. Users of the Internet PKI are people and processes that use client software and are the subjects named in certificates. These uses include

readers and writers of electronic mail, the clients for WWW browsers, WWW servers, and the key manager for IPSec within a router. This profile recognizes the limitations of the platforms these users employ and the limitations in sophistication and attentiveness of the users themselves. This manifests itself in minimal user configuration responsibility (e.g., trusted CA keys, rules), explicit platform usage constraints within the certificate; certification path constraints that shield the user from many malicious actions, and applications that sensibly automate validation functions.

What this means essentially, is that instead of keeping digital certificates in one centralized location, which makes them vulnerable to being hacked and tampered with, they would be spread out in a world-wide ledger, one fundamentally impervious to alteration. A hacker attempting to modify certificates would be unable to pull off such a fraud, as it would mean changing data stored on enumerable diversified blocks spread out across the cyber sphere.

Decentralization has already been proven as a highly effective way of protecting recorded data from tampering. Similarly, using a blockchain-type system to replace the single entity Certificate Authority, can keep our digital certificates much safer. It is in fact one of the only foreseeable solutions to keep the quantum revolution from undermining the foundation of PKI.

The study claims the key advantages of using blockchain based PKI includes:

- High reliability and performance: each node holds a copy of consistent certificate data, supports multiple duplicates and parallel queries, and better data consistency.
- Low cost: automatic operation of block chain system, low maintenance cost; no payment needed for self signed certificate.

Design Methodology. The design of blockchain-based PKI solution is based voting platform is based on hybrid X.509 certificates that is Verifiable, Accurate, Secure, and Transparent, as shown in figure below. A certificate contains certain information on PKI environment in the extension fields. The value of extension fields is as follow:

• Subject Key identifier: holds the certificate’s owner identity.

• Blockchain name: holds the name of the blockchain platform. Currently, most systems use Ethereum public blockchain but we envision covering more platforms.

• CA key identifier: holds the smart contract address of the current CA, if this is CA certificate. For non-CA certificates this field is empty.

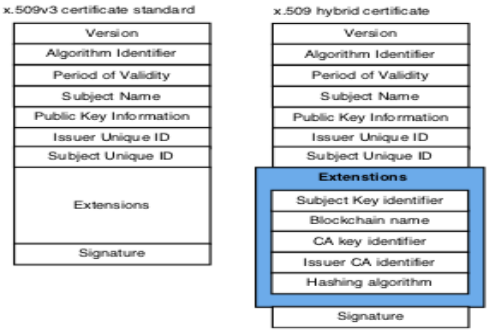
• Issuer CA identifier: holds the address of the smart contract of the CA, which issued this certificate. It allows validator to find parent CA smart contract in the blockchain and check if the certificate with the corresponding hash was issued and was not revoked. For root certificates this field is empty.

• Hashing algorithm: holds information regarding hashing algorithm, which used in calculation of the certificate's hash loaded into blockchain.

There are current PKI issues related to the study. For instance, two approaches were proposed to solve SSL/TLS PKI issues in current literature: Log-based PKIs schema, and decentralized networks of peer-to-peer certification, known as Web of Trust (WoT). Log-based PKIs approach has been proposed as an emerging solution to the problem of misbehaving CAs. The idea behind is using highly-available public log servers that monitor and publish the certificates issued by CAs. These public logs provide transparency by ensuring that only publicly-logged certificates are accepted and trusted by end-customers. Hence, users and servers will detect any misbehavior by CAs. Google's certificate Transparency (CT) is the most widely deployed log-based PKI, and it is currently available in both Chrome and Firefox. In parallel, many proposals intend to extend the features of Log-based PKI schema by support for revocation and error handling. Unfortunately, despite these benefits, log-based PKIs still have several challenges related for instance to certificate revocation such as the DoD OSCP primitives.

Web of Trust (WoT) approach is entirely decentralized: users are able to designate others as trustworthy by signing their public key certificates. By doing so, a user accumulates a certificate containing his public key and digital signatures from entities that have deemed him trustworthy. The certificate is then trusted by a third party if it is possible to verify that the certificate contains the signature of someone she/he trusts. This system benefits from its distributed nature because it removes any central point of failure. However, it makes it difficult for new or remote users to join the network, since some existing member of WoT typically must meet with the new user in person to have her/his identity verified and public key signed for the first time. Also, unlike a CA-based approach, the WoT has no way to deal with key revocation. A user is limited to choosing another user to be her/his "designated revoker" with the rights to revoke her/his certificate when the private key is lost or compromised. The current practical approaches depend on periodically pushing revocation lists to browsers.

However, such method makes invalid certificates to be trusted until the browser's revocation lists get the next push. Therefore, we propose to use the common standard DoD X.509v3 certificate with a minor addition to the extension fields with blockchain related information. Thus, the extended X.509 certificate can be validated by the classical DoD CA-based chain of trust or using blockchain-based PKI framework.

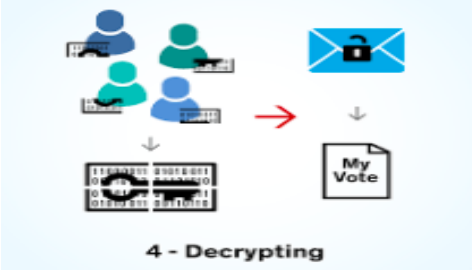


Key Advantages of Blockchain-based PKI voting platform. A blockchain-based PKI voting system has the following advantages over a traditional PKI. First, the validation of a certificate and its CA certificate chain is simple and fast. Second, blockchain- based PKI solves a longstanding problem of traditional PKIs by not requiring the use of a service that issues certificate revocation lists (CRLs) thanks to blockchain synchronization between network's nodes where any modification to the state of a certificate will be instantaneously notified to the all nodes.

Another important aspect in the context of Internet security is that the blockchain-based PKI provides flexible protection against the man-in-the-middle (MITM) attacks. Traditionally, MITM is considered as a major security risk implying attacker to hijack a browser's connection for a given websites by presenting a valid certificate (i.e., forged public-key) for that domain. For users and web browsers it is difficult to identify the replacement of certificate in case the related CA has been hacked by the attacker. Blockchain-based PKI approach makes MITM attacks virtually impossible as when a CA publishes or revokes the public key of a website/domain on the blockchain, the information will be distributed across thousand nodes, so tampering the public-key will be (theoretically) out of the question. Traditional PKI resolves MITM risks by embedding Root CA certificates into browser installations, thus artificially expanding CA entrance barriers and increasing the time necessary for Root CA certificate revocation.

The access rights to modify certificate authority data is based on user accounts system of the blockchain platform, the lost password results in irrevocably lost access to the account. Some solution of the troubles of CA loosing its blockchain access password could be the arranging empty smart contract and copying all the data from the old smart contract to the new one, as theoretically speaking any smart contract is always available for reading to anyone. Obviously, the

establishment of the new CA smart contract could result in reissue of CA certificate (at least in our current implementation), as CA certificates may contain the reference to the corresponding smart contract.



Current literature claims that a decentralized software update framework that eliminates single points of failure, enforces transparency, and provides efficient verifiability of integrity and authenticity for software-release processes. Independent witness servers collectively verify conformance of software updates to release policies, build verifiers validate the source-to-binary correspondence, and a tamper-proof release log stores collectively signed updates, thus ensuring that no release is accepted by clients before being widely disclosed and validated.

For example, the scalability of distributed ledgers (DLs), both in total transaction volume and the number of independent participants involved in processing them, is a major challenge to their mainstream adoption, especially when weighted against security and decentralization challenges. Most existing distributed ledgers are unable to “scale-out” – growing total processing capacity with number of participants – and those that do compromise security or decentralization. Distributed ledgers derive current system state from a blockchain, or a sequence of totally-ordered blocks containing transactions.

The fundamental strength of any blockchain solution is cryptographic security. Pertinent examples suggest that Skipchains enable software clients to efficiently navigate arbitrarily long blockchain timelines both forward and backward, providing proof of transaction validity without the need for a full record of the blockchain. Back-pointers in Skipchains are cryptographic hashes, while forward-pointers are collective signatures by a group of witnesses. Skipchains are a

useful cryptographic blockchain structure loosely inspired by skip lists. The fundamental concept of a skip list is to augment a conventional singly linked or doubly linked list with additional long-distance links, which are structurally redundant but allow much more efficient traversal and search across arbitrary distances along the timeline in a logarithmic, rather than linear, number of steps. In this way, our software can validate a referenced block by using cryptographically validated markers that represent a large group of adjacent blocks. The end result is that even resource-constrained clients, such as those on mobile phones, can obtain and efficiently validate binary updates using a hard-coded initial software version as a trust anchor. Blockchain is not a panacea, however, with encryption that exceeds most governments and financial institutions, the study will provide a suitable blockchain-enabled online voting system that distributes recorded ballots on a ledger distributed across DoD networks and the Internet of trusted voting authority partners. The implementation of this blockchain voting framework will afford a level of data consistency and security many orders of magnitude superior to ordinary data storage and retrieval solutions and beyond the standards widely held today.

Integrity and authenticity are essential. Having proper protection for signing keys to defend against such single points of failure is therefore a top priority. Second, the lack of transparency mechanisms in the current infrastructure of software distribution leaves room for equivocation and stealthy back-dooring of updates by compromised, coerced, or malicious software vendors and distributors.

The study security goals are as follows:

1. No single point of failure: The software-update system should retain its security guarantees in case any single one of its components fails (or gets compromised), whether it is a device or a human.

2. Source-to-binary affirmation: The software-update system should provide a high assurance-level to its clients that the deployed binaries have been built from trustworthy and untampered source code.

3. Efficient release-search and verifiability: The software-update system should provide means to its clients to find software release (the latest or older ones) and verify its validity in an efficient manner.

4. Linear immutable public release history: The software-update system should provide a globally consistent tamper-evident public log where each software release corresponds to a unique log entry that, once created, cannot be modified or deleted.

5. Evolution of signing keys: The software-update system should enable the rotation of authoritative keys, even when a (non-majority) subset of the keys is compromised.

6. Timeliness of updates: Clients should be able to verify that the software indeed corresponds to the latest one available.

System Model. Developers write the source code of a software project and are responsible for approving and announcing new project releases. Each release includes source code, binaries (potentially, for multiple target architectures), and metadata such as release description. A snapshot refers to a set of releases of different software projects at a certain point in time. Projects can have single or multiple packages. Witnesses are servers that can validate and attest statements. They are chosen by the developers and should be operated ideally by both developers and independent trusted third parties. Witnesses are trusted as a group but not individually. Build verifiers are a subset of the witnesses who execute, in addition to their regular witness tasks; reproducible building of new software releases and compares them to the release binaries. Witnesses and build verifiers jointly form an update cothority (collective authority). The update timeline refers to a public log that keeps track of the authoritative signing keys, as well as the software releases. Users are clients of the system; they receive software releases through a (untrusted) software-update center.

An attack on the system is successful if an attacker manages to accomplish at least one of the following:

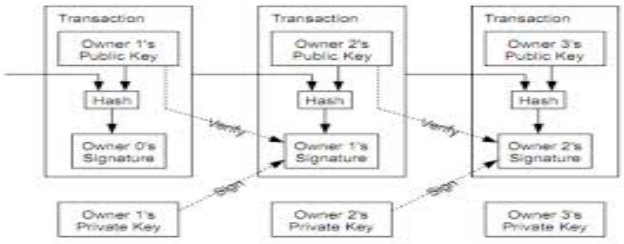
- Make developers sign the source code that they do not approve.
- Substitute a release binary with its tampered version such that the update cothority signs it.
- Trick the update cothority into signing a release that is not approved by the developers.
- Create a valid fork of the public release history or modify/revoke its entries; or present different users with different views of the history.
- Trick an outdated client into accepting a bogus public key as a new signing key of the update cothority.

- Get a client to load and run a release binary that is not approved by the developers or validated by the update cothority.

Specific protocols for the system model. How does a “light” (low-power, mobile device) client securely confirm a thing is on the blockchain? Software-update mechanisms are critical to the security of modern systems, but their typically centralized design presents a lucrative and frequently attacked target. O(log N) skipchains are authenticated data structures that combine ideas from blockchains and skiplists. This confirms a cryptographically traversable blockchain with forward pointers that include signing-key-group deltas. Therefore you have Backward and Forward Verifiability of the Blockchain. Skipchains enable clients (1) to securely traverse the timeline in both forward and backward directions and (2) to efficiently traverse short or long distances by employing multi-hop links. Backward links are cryptographic hashes of past blocks, as in regular blockchains. Forward links are cryptographic signatures of future blocks, which are added retroactively when the target block appears.

Each time a consensus group signs a forward link at one level, all consensus group members derive new key pairs for the next level and include the next-level public keys in the metadata for the forward link currently being signed. Once the collective signature at this level is formed, all honest consensus group members securely erase their private keys at this level and retain only the private keys for the next level. In this way, a powerful attacker who can compromise a threshold number of private keys over a long time period obtains only the ability to forge comparably long-distance forward links. All shorter-distance forward links remain secure and unforgeable. Such an attacker then can only compromise devices that sync with the blockchain extremely rarely. Devices that sync more regularly, either by going online or via peer-to-peer updates, remain immune to such an attacker because they only ever follow shorter-distance forward links.

We distinguish randomized and deterministic skipchains, which differ in the way the lengths of multi-hop links are determined. The link length is tied to the height parameter of a block that is computed during block creation, either randomly in randomized skipchains or via a fixed formula in deterministic skipchains. In both approaches, skipchains enable logarithmic-cost timeline traversal, both forward and backward.



First, skipchains enable clients to securely and efficiently traverse arbitrarily long timelines, both forward and backward from any reference point. If the client has

the correct hash of an existing block and wants to obtain a future or past block in the timeline from an untrusted source (such as a software update server or a nearby peer), to cryptographically validate the target block (and all links leading to it), the client needs to download only a logarithmic number of additional, intermediate blocks.

Secondly, suppose two resource-constrained clients have two reference points on a skipchain, but have no access to a database containing the full skipchain, *e.g.*, clients exchanging peer-to-peer software updates while disconnected from any central update server. Provided these clients have cached a logarithmic number of additional blocks with their respective reference points – specifically the reference points' next and prior blocks at each level – then the two clients have all the information they need to cryptographically validate each others' reference points. For software updates, forward validation is important when an out-of-date client obtains a newer up- date from a peer. Reverse validation (via hashes) is useful for secure version rollback, or in other applications, such as efficiently verifying a historical payment on a skipchain for a cryptocurrency.

Each block in the Skipchain consists of the following data elements:

- Root hash of the Merkle tree containing all transactions in the current block;
- Root hash of the Merkle tree representing the entire Skipchain's current state;
- Hash of the current block, which acts as a unique identifier for the current block;
- Hash backward link pointing to the previous block;
- List of forward and backward links pointing to different blocks in the Skipchain for quick navigation within the chain;
- List of Cothority nodes responsible for handling that block.

One major innovation of this technology is that it allows participants to store and transfer data across the Internet without the need for a centralized third party. Data stored on a decentralized blockchain is immutable to changes, making the blockchain a trustworthy source of data.

Ethereum. A possible an open source, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality

that can add value to the proposed study is Ethereum. The entire ecosystem of Ethereum works on the basis of smart contracts. For the uninitiated, smart contracts are basically how things get done in the Ethereum eco-system. To put it in layman terms, smart contracts are automated contracts that enforce and facilitate the terms of the contract itself. Ethereum might not be a good idea but I like it. Ethereum has made a spectacular comeback from an absolute disaster, and it looks like it's going fulfill all the expectations that people had had in it when it started. More than anything, the true power of Ethereum lies in its full scope. It is not just a currency; it is a platform on which people can build projects that will dictate the future. If decentralization is indeed the future, then Ethereum is going to be in the front and center of it.

Pros

- Is growing at an exponential pace.
- Has the majority of the original big dogs who have created Ethereum in its corner.
- Has reversed the DAO hack and given back the stolen money to its rightful owners (the DAO token holders).
- Is being constantly updated with the latest changes.
- Has a higher hash-rate than ETC.
- A powerful example of what the Ethereum community is capable of when it comes together to solve a problem.
- ETH is backed by a powerful group of over 200 corporations called the Enterprise Ethereum Alliance (EEA), which aims to use the blockchain technology to run smart contracts at Fortune 500 companies. Members include: Microsoft, JP Morgan, Toyota, ING, etc.

Cons

- Goes against the policy of immutability.

An open source programming language the study proposes to look into is Vyper. It is a general-purpose, experimental programming language that compiles down to EVM (Ethereum Virtual Machine) bytecode, as does Solidity. However, Vyper is designed to massively simplify the process in order to create easier-to-understand Smart Contracts that are more transparent for all parties involved, and have fewer points of entry for an attack. Then you need some sort of a decentralized release-approval.

The first step towards this kind of architecture involves enlarging the trust base that approves software releases. Instead of using a single (shared) key to sign updates, each software developer signs using their individual keys. At the beginning of a project, the developers collect all their public keys in a *policy* file, together with a threshold value that specifies the minimal number of valid

developer signatures required to make a release valid. Complying with our threat model, we assume that this policy file, as a trust anchor, is obtained securely by users at the initial acquisition of the software, *e.g.*, it can reside on a project's website as often is the case with a single signing key in the current software model.

Upon the announcement of a software release, which can be done by a subset or all developers depending on the project structure, all the developers check the source code and, if they approve, they sign the hash of it with their individual keys, e.g., using PGP (message format) and forward secrecy, and they add the signatures to an append-only list. Signing source code, instead of binaries, ensures that developers can realistically verify (human-readable) code. The study will claim that forward secrecy get baked into the protocol.

The combination of the source code and the signature list is then pushed to the software-update center from where a user can download it. For simplicity, we first assume that the update center is trusted, later relaxing this assumption. When a user receives an update, she verifies that a threshold of the developers' signatures is valid, as specified in the policy file already stored on user's machine. If so, the user builds the binary from the obtained source code and installs it. An attacker trying to forge a valid software-release needs to control the threshold of the developers' keys, which is presumably harder than gaining control over any single signing key.

Digital identities. A challenge in any election event revolves around the validation of voter identities and voting eligibility. Even in traditional paper-based voting systems, it can be difficult to identify that voters are who they claim to be. For example, some ineligible voters have been allowed to cast votes remotely by registering under the names of deceased individuals. In the digital context, the verification of a voter is fundamentally more challenging due to ongoing limitations around securely binding physical and digital identities. Having voter identities dispensed and revoked by central authorities puts voters back at the mercy of a few administrators who can decide which votes count. The role of validators, meanwhile, is reduced to auditing for fraudulent votes, which can be achieved far more simply.

More specific issues are the variants of security using credentials. The DoD uses the CAC. But the CAC is going to be replaced. The study will suggest a set of smart security features:

- Use of the latest cryptography to secure the data
- COTS developed end-to-end applications
- Biometric data for secure identification
- Highest level of privacy and security
- Confidential communication by contact or contactless, or both (dual interface)

A set of secure application features:

- Secure data storage, identification, authentication and digital signature
- Protection by PIN or biometric authentication
- Multi-usage of electronic cards for identification, travel, payment and much more
- Automated verification

Current findings in various platforms suggest that there are three groups of security features that can be implemented:

1. Security Features that are difficult to copy

- Security printing
- Optically variable features
- Non-printed security - elements such as holograms and surface structures

2. Security Features that are difficult to produce

- Security printing
- Optical effects
- Surface structures

3. Technologies/materials that are difficult to come by

- Certain printing technologies
- Security paper
- Special holograms
- Special high-security inks

The DoD, NIST and NSA verify that digital security is the 5th level the study needs to mention. It's a dimension unseen by the naked eye but real. Think about this for a moment. Electronic identity documents such as national ID cards

or ePassports equipped with a microprocessor and secure applications offer an additional dimension of security to the physical features. Why?

Because secured digital data is virtually impossible to alter and allows reliable automated verification and easy machine readability. Our current election infrastructure does not earn the trust of those using it, rather, it demands it without offering the verifiable proof that a voter's vote was cast and counted as intended. Voters must be provided sufficiently convincing evidence that election integrity has not been compromised. CABIS offers a wide array of features and functions for processing tenprints, palm prints, finger latents, palm latents, faces, and irises. As one of the most accurate systems in the world, CABIS ensures service resiliency while providing information safety through the use of built-in safeguards such as fault tolerant architecture, disk mirroring, automated database backups, and high availability options.

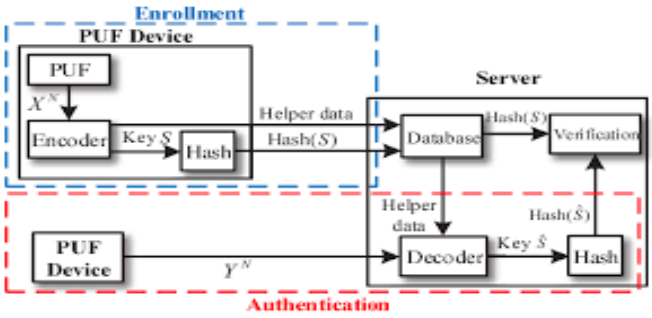
- Accuracy: Superior algorithms and comprehensive set of powerful tools that analyze and enhance the quality of prints increasing the probability of a hit.
- Speed: While transactions are processed in the background, users can concurrently perform multiple tasks swiftly and precisely.
- Flexibility: Operates in a multitasking environment and is built for expansion as demands and upgrade requirements arise.
- Scalability: Manages cost effective growth in response to increasing customer demands.
- Interoperability: Non-proprietary NIST record formats are used in database storage. As a result, the system can be interfaced with external ABIS systems, criminal history systems, Livescans, mobile devices, web-based solutions, as well as other information systems.

As we replace the CAC, we can look into the secure national polycarbonate e-ID card includes an embedded PKI blockchain application, which enables online authentication and digital signature with electronic certificates. The Java-based technology offers multi-application capabilities. A contactless chip in the card allows for special functions such as a fingerprint check and can be utilized when traveling. We believe, been asking the right questions:

- Why can't voter registration be seamless? Identification and authentication

- Why should voting be linked to polling places? People are mobile. They should be able to vote anywhere
- Why do they have to require that older or disabled people be wheeled into a polling place?
- Why can't they vote at home?

Key management. Cryptography like mathematics starts off with assumptions (axioms) and from those assumptions derives conclusions. Unlike traditional approaches that depend on asymmetric key cryptography, a secure key management solution should use only hash-function cryptography, allowing verification to rely only on the security of hash-functions and the availability of the history of cryptologically linked root hashes (the secure voting PKI blockchain). Therefore, that is, the study proposes to redefine the word "blockchain" to mean "Merkle tree" but only for crypto key management purposes. Through the properties of verifiable authenticity, identity of the client, and non-global positioning system-based non-spoofable time; the study will provide a better alternative to conventional key management systems with provenance, integrity and identity associated with digital assets. This implementation consumes far less storage and bandwidth than widely proliferated blockchain technology and can provide the above defined attributes for thousands of files a second scalable to billions. Authentication mechanisms can no longer be based on simply stored secret data or PKI as the secret keys could get stolen or leaked. This issue can be addressed by using the physical properties of a device (the device fingerprint) as the unique secret keys for authentication purposes.



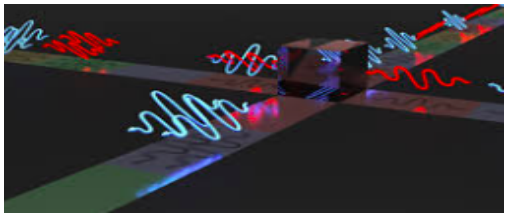
For example, the study proposes to glean from new advances in physical properties, which are unpredictable and unclonable due to un-controllable process variations during manufacturing. The most secure, robust and deployed PUF is the SRAM PUF. Hence, SRAM PUF based IDs are unique per device and can be used for authentication in addition to aiding the creation of unique cryptographic keys. Physical Unclonable Functions or (PUFs) use device unique random patterns to differentiate chips from each other. PUFs, and SRAM PUFs in particular, are designed to be impossible to duplicate, clone or predict. This makes them very suitable for applications such as secure key generation and storage, device authentication, flexible key provisioning and chip asset management. PUFs are actively stimulated and executed to exploit the random-ness in their behavior. A good way to look at a PUF is as a device fingerprint.

The noisy behavior of this device fingerprint is also utilized to the advantage of the system. The noise entropy is harvested to create strong, independent random numbers with high entropy. Strong independent random number generators are needed in all kinds of cryptographic protocols and are often the weakest link in a cryptographic implementation.

$$E_H(\rho) \le \left(\frac{d_\rho}{\mathrm{Tr}(\Pi_H^\rho)}\right)^2 \left(\overline{\chi}_{\overline{S}}(H) + |S \cap I(\rho \otimes \rho^*)| \frac{d_S^2}{d_\rho}\right).$$

$$\frac{\mathrm{Tr}(\Pi_H^\rho)}{d_\rho} = \frac{1}{|H|}\left(1 + \sum_{h \in H\setminus\{1\}} \frac{\chi_\rho(h)}{d_\rho}\right) \ge \frac{1}{|H|} - \overline{\chi}_\rho(H) \ge \frac{1}{2|H|}.$$

The matter of key protection and key governance in blockchain is a crucial yet mostly unresolved obstacle. This proposal presents the basic research challenges that need to be accomplished to adapt traditional PKI technologies to blockchains, simplifying both identity management and key management for individuals and institutions while at the same time enhancing both security and privacy. One example claims in a keyless signature system, the functions of signer identification — and of evidence integrity protection — are separated and delegated to cryptographic tools suitable for those functions. For example, signer identification may still be done by using asymmetric cryptography but the integrity of the signature is protected by using keyless cryptography — the so-called one-way collision-free hash functions, which are public standard transformations that do not involve any secret keys. The main goal of digital time stamping is to prove that electronic data-items were created or registered at a certain time. A simple method is to use a trusted service (with a precise clock) that provides data items with current time value and digitally signs them. The assumption of unconditionally trusted service hides a risk of possible collusions that may not be acceptable in applications. The risks are especially high in centralized applications like electronic patent- or tax filing as well as in electronic voting, where the possible collusions are related to direct monetary (or even political) interests.



Overarching vulnerabilities in current classical crypto systems. Most current web authentication solutions use a login form, which sends the username

and password to the server as a HTTP request. In most cases, the password is sent in plain text; sometimes, this plain text password is sent through an SSL connection. On the server, the password is hashed and compared to a stored hash. Another vulnerability of the classical scheme is replay attacks, where an eavesdropper captures valid authentication credentials over a network and replays them to the server at a later time to gain access.

Most systems also use a process called salting in which random bits are added to the end of the password before it is hashed to prevent attacks through pre-computed hash tables. There are several major attack vectors for this current system. The first major vector is brute-force hash cracking. In this attack, the adversary has gained access to the hash of the password, usually by compromising the server storing the password hashes. Once the adversary has access to the password hashes, he hashes many common passwords to see which hash matches and thus determines the original plaintext password. A user can protect against this attack by choosing a password that is difficult to guess, and the server can protect against attacks. A second possible attack is wire sniffing, where a malicious entity listens in on the client's connection to the server and reads the password that's sent over the network. Most large sites use SSL for authentication, which encrypts this connection and makes it more difficult for listeners to read any data; however, some still do not. Finally, the server itself could behave maliciously. Since passwords are almost always sent to the server without hashing, the server can see all passwords in plaintext.

In order to ameliorate these threats, the study proposes and suggests that a randomized authentication system based on canonical interactive proof protocol called 'zero knowledge proof' to be studied and validated as a possible application to authentication and identity. This system never transmits information that can be used to easily recover a user's password; furthermore, each login attempt will be different, resulting in a system that is, overall, more secure. Hopefully, the conclusion is that simplicity and ease of use for the client demonstrate that, with appropriate setup, a zero knowledge authentication protocol can be feasibly implemented for websites without detriment to the user experience. The results would include adding a random salt to the server side public key storage, investigating the existing libraries for JavaScript cryptography and big numbers for correctness, integrity, and security of cryptographic functions.

Based on the above assumptions, the study proposes a new authentication and identity protocol approach called "Zero Knowledge Proof" by which if one could come up with a computer program (for the Verifier) that extracts useful information after participating in a run of the protocol, then it would be possible to use a 'time machine' on that program in order to make it extract the same amount of useful information from a 'fake' run of the protocol where the Prover doesn't put in any information to begin with. A Zero Knowledge Proof has an interaction between two computer programs (or Turing machines) — respectively

called a Prover and a Verifier — where the Prover works to convince the Verifier that some mathematical statement is true.

In the course of the study discussion, one can describe three critical properties that any zero knowledge proof must satisfy:

- Completeness: If the Prover is honest, then she will eventually convince the Verifier.
- Soundness: The Prover can only convince the Verifier if the statement is true.
- Zero-knowledge(ness): The Verifier learns no information beyond the fact that the statement is true.

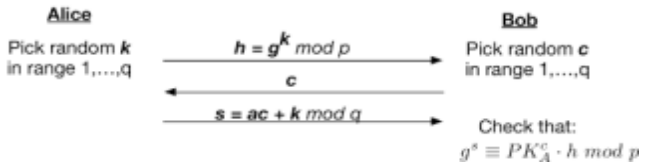
The real challenge for this concept turns out to be finding a way to formally define the last property. How does the assumption state that a Verifier learns *nothing* beyond the truth of a statement? The study would verify that the Zero-knowledge protocols enable the transfer of assets across a distributed, peer-to-peer blockchain network with complete privacy. In regular blockchain transactions, when an asset is sent from one party to another, the details of that transaction are visible to every other party in the network. By contrast, in a zero knowledge transaction, the others only know that a valid transaction has taken place, but nothing about the sender, recipient, asset class and quantity. The identity and amount being spent can remain hidden, and problems such as “front-running” can be avoided.

The study is not necessarily concerned about digital signatures in the classical sense, however. The concern was with *identification.* Specifically, the assumption imagines that Alice has published her public key to the world, and later on wants to prove that she knows the secret key corresponding to that public key. This is the exact problem in real-world protocols such as public-key SSH, so it turns out to be a well-motivated study prospect.

The identification protocol begins with the assumption that the public key would be of a very specific format. Specifically, let *p* be some prime number, and let *g* be a generator of a cyclic group of prime-order *q*. To generate a keypair, Alice would first pick a random integer *a* between 1 and *q*, and then compute the keypair as:

$$PK_A = g^a \ mod \ p, \ SK_A = a$$

Alice keeps her secret key to herself, but she’s free to publish her public key to the world. Later on, when she wants to prove *knowledge* of her secret key, she conducts the following simple interactive protocol with Bob:



First, the assumption should ask if the protocol is *complete.* This is the easiest property to verify: if Alice performs the protocol honestly, should Bob be satisfied at the end of it? In this case, completeness is pretty easy to see just by doing a bit of substitution:

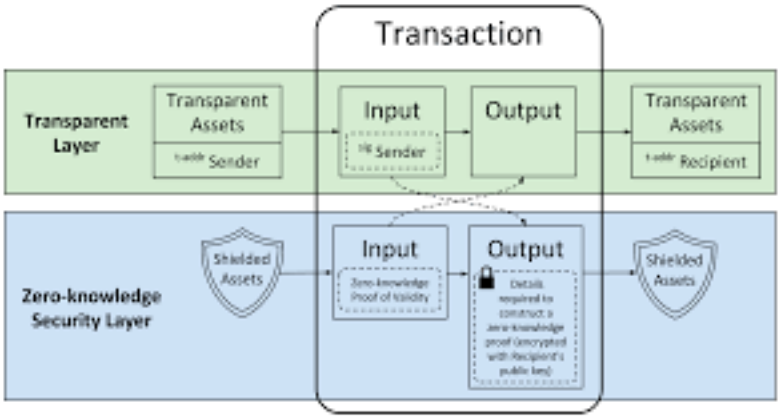
$$\begin{aligned} g^s &\equiv PK_A^c \cdot h \mod p \\ g^{ac+k} &\equiv (g^a)^c \cdot g^k \mod p \\ g^{ac+k} &\equiv g^{ac+k} \mod p \end{aligned}$$

An example of this concept is to prove knowledge of a secret a for some public key $g^a \ mod \ p$— but we don't actually know the value a. Our study Simulator assumes that the Verifier will choose some value c as its challenge, and moreover, it knows that the honest Verifier will choose the value c only based on its random number generator — and not based on any inputs the Prover has provided:

- First, output some initial g^{k_1} as the Prover's first message*,* and find out what challenge c the Verifier chooses.
- *Rewind the Verifier*, and pick a random integer z in the range $\{0, \ldots, q-1\}$.
- Compute $g^{k_2} = g^z * g^{a(-c)}$ and output g^{k_2} as the Prover's new initial message.
- When the Verifier challenges on c again, output z.

Notice that the transcript g^k, c, z will verify correctly as a perfectly valid, well-distributed proof of knowledge of the value a. The Verifier will accept this output as a valid proof of knowledge of a, even though the proposed study Simulator does not know a in the first place.

What this proves is that if we can rewind a Verifier, then one can always trick the Verifier into believing there is knowledge of a value, even when we don't. And since the statistical distribution of our protocol is identical to the real protocol, this means that our protocol must be zero knowledge — against an honest Verifier.



Zero-knowledge in this case means that no one besides you has the keys to your data, not even the service you're storing your files with. Also known as private encryption, it is the ultimate way in which you can keep your data private, though it does come with a few downsides: most important of these is that if you lose your password, it is gone forever. It allows a party to prove that he/she knows something (i.e. credential), without having to send over the value of the credential. In this implementation, it will be used to prove the password of the user without sending over the actual password. The system also allows for no password hashes to be stored on the server. For example, a server running a zero knowledge web authentication system would provide additional resilience against an adversary listening in on server-client messages. Furthermore, the probabilistic nature of zero knowledge proofs is not an issue; in particular, even current client-authentication schemes are not perfectly error-free, as the adversary can simply guess the right password by chance. This difference between zero knowledge proofs and traditional interactive proofs mirrors the difference between zero knowledge authentication and current login systems. A zero knowledge authentication scheme would encode the user's identity (e.g. username and password) as a hard problem, and use an interactive zero knowledge proof model to authenticate.

Software Engineering. The engineering of any software system can be roughly divided into four stages: specification, implementation, verification/validation, and maintenance. These stages are not mutually exclusive; for example, a specification can be refined during implementation and verification/validation can occur continuously during implementation and maintenance.

Specification is the most important stage, particularly when building critical systems: it informs the entire development process by determining what must be built and what it must, and must not, do. Even if a system works perfectly, it is impossible to provide convincing evidence of its correctness without using and consistently maintaining a high-quality, accurate specification throughout the development process. Such evidence is essential for critical systems to which we entrust our lives, money, or votes.

Implementation is the transformation, either automatic or manual (typically some of each), of a system's specification into executable code. The quality of an implementation is heavily influenced by choices of programming languages, coding conventions, and supporting tools, some of which are driven by choices made during specification.

Verification and validation are closely related, but independent, procedures. Verification is an evaluation of whether the implementation is a correct realization of its specification, while validation is an evaluation of whether the implementation is satisfactory to external stakeholders. Verification and validation are typically carried out continuously and automatically during the development process, using a variety of tools and techniques; the results of verification and validation are the primary sources of evidence for the implementation's correctness.

Maintenance includes tracking and fixing defects discovered in the implementation, modifying the specification and implementation as required by technical issues or feedback from external stakeholders, and adapting the implementation to new hardware and software platforms and new deployment environments. Maintenance is closely linked with verification and validation, since new evidence of the system's correctness must be provided when changes are made to either its specification or its implementation.

Testing. Software testing practices are a key component of any software engineering methodology. Even when parts of a system are formally verified, testing can provide additional assurance that the system satisfies its requirements, behaves as expected for particular inputs, works correctly in diverse environments, and has sufficient performance.

Testing serves the key functions of uncovering flaws quickly and ensuring that previously fixed flaws do not recur in later software versions. It is impossible for testing to reveal all possible flaws in any realistic program—the number of possible inputs for any such system is so large that an exhaustive test would effectively run forever—but tests that successfully reveal and prevent recurrences of some flaws provide some evidence for a system's correctness.

Verification and testing should not be viewed as opposing alternatives, but rather as complementary techniques that together provide assurance in the developed software. It is not feasible to exhaustively test every possible input and every possible path through any non-trivial program, while verification offers guarantees over all possible inputs. However, verification usually cannot scale to provide those guarantees for entire systems; verification tools must sometimes make simplifying assumptions about the environment in which software runs or reason about a simplified model of the actual system. Since testing can exercise the real system in a real environment, it can uncover flaws that are beyond the scope or capability of verification.

Different testing practices provide complementary types of assurance; no single testing practice is sufficient. Instead, multiple types of testing should be used in combination on every project.

Configuration Management. Complex software systems may depend on many components outside their own implementations, including database servers, application servers, web servers, authentication systems, etc. They may also need to run on several different target platforms with different operating systems and capabilities, or on multiple commercial cloud infrastructures with different deployment and management processes.

Implementation Language. The choice of implementation language is clearly important, and there are many possible choices; hundreds (possibly thousands) of programming languages exist and dozens of those are actually viable, with widespread adoption, tool support, and support communities. In most cases, language choice determines programming style: for example, Eiffel imposes an pure object-oriented style while Haskell imposes a pure functional style. Language choice also determines the set of existing functionality, in the form of standard libraries accompanying the language or well-established external libraries available for use with the language, on which developers can rely while building the implementation.

Implementation languages are distinguished by different styles, different methods for error handling, different security guarantees, and different ways in which programmers can make mistakes (both minor and catastrophic). For rigorous software engineering, we generally recommend languages with strong type systems that actively prevent programmers from making any of a large class of errors. We also recommend languages that automatically handle memory allocation and deallocation, which prevents another large class of errors and many potential security issues. Finally, we recommend languages that either have good specification and verification tool support or are designed explicitly for the implementation of high-assurance software. It is certainly possible to implement reliable software in languages that do not have all these features—for example, code can be written in a safe subset of C, specified with ACSL, and verified with various tools—but it is significantly more difficult to do so.

Roots of Trust. As previously mentioned, all computing systems have multiple layers of abstraction; the lowest layer is the hardware on which the system runs, followed by the firmware, the operating system (which may itself have multiple layers), and finally the application software. In general, higher layers of the system must trust that lower layers are not malicious and are performing according to their specifications. The *roots of trust* in a computing system, also the hardware and software components of the system that are inherently trusted.

For example, in current general-purpose computers, the boot firmware—the first code that executes when the machine is powered on—is a root of trust. If the boot firmware is secure, the system can use it to "bootstrap" a chain of trust; for example, the system can use trusted cryptographic functions to verify the integrity of software modules before loading them, and the loaded modules can then be trusted to perform other functions. However, if the boot firmware is compromised in some way, it can inject malicious code into any software that it loads, and as a result none of the system can be trusted. It is therefore critical to ensure that the boot firmware, and any other roots of trust in a system, is protected from tampering.

Currently, the only available way to ensure the integrity of a system's roots of trust is by using a piece of dedicated hardware called a Trusted Platform Module (TPM). A TPM can check the integrity of the device's boot firmware before allowing it to run, and can also provide secure storage for encryption keys to protect the contents of a machine's disk and authenticate authorized users before allowing the machine to boot. By design, the integrity of a TPM can only be compromised by a direct attack on its hardware such as physical delamination of the TPM chip or direct measurement of its radiation output. Thus, if the hardware has not been physically tampered with, the TPM and the functionality embedded within it can be considered secure and used to bootstrap trust at higher levels of abstraction.

TPM functionality is embedded into many of today's computing systems. However, the availability of TPM functionality is not enough; the layers of software above a TPM must use it properly in order to provide any assurance. Therefore, when building an PKI Blockchain voting system, it is essential that the roots of trust of the system be explicitly enumerated and that the chain of trust for each originates in secure hardware.

Post Quantum Era. The study proposes the implications of the post-quantum computing world. Most worry about the people and organizations that rely on cryptographic codes to protect sensitive information. When the first decent-sized quantum computer is switched on, previously secure codes such as the commonly used RSA algorithm will become instantly breakable; hence a blockchain PKI system. First, let's a make a distinction between symmetric and asymmetric codes. Symmetric codes use identical keys for encrypting and decrypting a message. Quantum computers can dramatically speed up an attack against these kinds of codes. However, symmetric codes have some protection. Doubling the size of the key counteracts this speed up. So it is possible for code makers to stay ahead of the breakers, at least in theory.

Asymmetric codes use different keys for encrypting and decrypting messages. In so-called public key encryption systems such as the popular RSA algorithm, a public key is available to anyone who can use it to encrypt a message. But only those with a private key can decrypt the messages and this, of course, is kept

secret. The security of these systems relies on so-called trap door functions: mathematical steps that are easy to make in one direction but hard to do in the other. The most famous example is multiplication. It is easy to multiply two numbers together to get a third but hard to start with the third number and work out which two generated it, a process called factorization.

The problem of factorization is to find a number that divides exactly into another. Mathematicians do this using the idea of periodicity: a mathematical object with exactly the right periodicity should divide the number exactly, any others will not. One way to study periodicity in the classical world is to use Fourier analysis, which can break down a signal into its component waves. The quantum analogue to this is the quantum Fourier sampling and Shor's triumph was to find a way to use this idea to find the periodicity of the mathematical object that reveals the factors. Thanks to Shor, any code that relies on this kind of asymmetry (i.e., almost all popular public key encryption systems) can be cracked using a quantum Fourier attack. One must always evaluate the cost of the protection against the cost associated with the loss of your data. But part of that evaluation must include the certainty of the security solution. Therefore, a minimal part of the study evaluation must include the certainty of the security solution. Consequently, the search for algorithms believed to be resistant to attacks from both classical and quantum computers have focused on public key algorithms. It is unclear when scalable quantum computers will be available. However, in the past year or so, researchers working on building a quantum computer have estimated that it is likely that a quantum computer capable of breaking 2000-bit RSA in a matter of hours could be built by 2030 for a budget of about a billion dollars. Thus, transitioning from 112 to 128 (or higher) bits of security is perhaps less urgent than transitioning from existing cryptosystems to post-quantum cryptosystems.

This post-quantum transition raises many fundamental challenges. Unfortunately, the bits-of-security paradigm does not take into account the security of algorithms against quantum cryptanalysis, so it is inadequate to guide our transition to quantum-resistant cryptography. There is not yet a consensus view on what key lengths will provide acceptable levels of security against quantum attacks. For symmetric key systems, one simple heuristic is to double the key lengths to compensate for the quadratic speedup achieved by Grover's algorithm. But this recommendation may be overly conservative, as quantum computing hardware will likely be more expensive to build than classical hardware. This study will propose appropriate algorithms, as the replacements for currently standardized public key algorithms are not yet ready, a focus on maintaining crypto agility is imperative. Until new quantum-resistant algorithms are standardized, agencies should continue to use the recommended algorithms currently specified in NIST standards.

The main thrust of the study, in this case, is the focus on crypto agility. Many of our most crucial communication protocols rely principally on three core cryptographic functionalities: public key encryption, digital signatures, and key exchange. The security of these depends on the difficulty of certain number theoretic problems such as Integer Factorization or the Discrete Log Problem over various groups. For arguments sake, two types of authentication services can be provided using cryptography: integrity authentication and source authentication:

• An integrity authentication service is used to verify that data has not been modified, i.e., this service provides integrity protection.
• A source authentication service is used to verify the identity of the user or system that created information (e.g., a transaction or message).

Key establishment protocols use key-establishment schemes in order to specify the processing necessary to establish a key. However, key-establishment protocols also specify message flow and format. Key-establishment protocols need to be carefully designed to not give secret information to a potential attacker. The study will focus on the application of virtual crypto currencies tied to blockchains and how key are used. For example, In general, a single key shall be used for only one purpose (e.g., encryption, integrity authentication, key wrapping, random bit generation, or digital signatures). There are several reasons for this:

1. The use of the same key for two different cryptographic processes may weaken the security provided by one or both of the processes.
2. Limiting the use of a key limits the damage that could be done if the key is compromised.

3. Some uses of keys interfere with each other. For example, consider a key pair used for both key transport and digital signatures. In this case, the private key is used as both a private key-transport key to decrypt the encrypted keys and as a private signature key to apply digital signatures.

This principle does not preclude using a single key in cases where the same process can provide multiple services. In addition to using the key, in bitcoin there's a process of seeding a private key per transaction in order to preserve user anonymity, so multiple private keys are generated from the same seed. Thus, the seed is highly sensitive and must remain secure. This means that the seeding process must be protected within boundaries of the secure hardware and not just the signing operation. Most secure hardware cannot give protection to both processes without massive modification: (i) derivation of private key from seed, (ii) the private key itself used to sign the transaction. In order to prevent hackers from decrypting huge masses of restricted information (often including a lion's share of customer's PII data), there is a need first to prevent the key compromise, because otherwise the attacker can decrypt the information at will. In addition, one must circumvent out-of-ordinary usage key usage. Crypto anchors are all about preventing the first possibility from taking place, and monitoring and/or proactively preventing the second one. The study is not suggesting that one uses Bitcoin but a similar seeding capability for the blockchain key management platform. All cryptographic information requires integrity protection. The best that can be done is to use reasonable measures to prevent modifications, to use methods to detect any modifications that occur (with a very high probability), and to restore the information to its original content when unauthorized modifications have been detected.

The fly in the ointment.

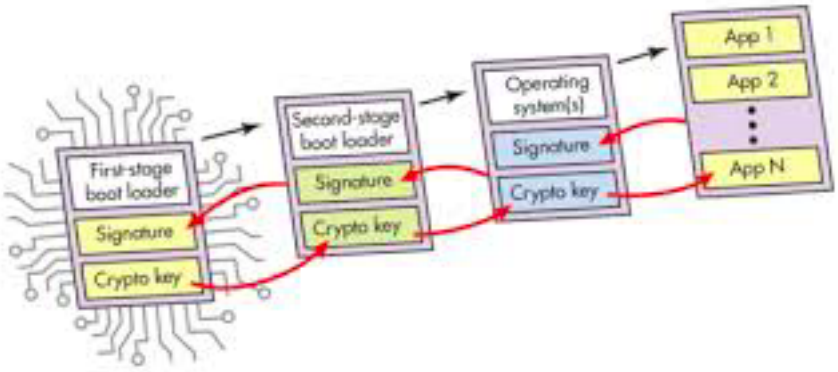
Cryptographers have spent decades advocating for their preferred solutions to those challenges—a suite of techniques known as "end-to-end verifiable voting." These techniques make no use of blockchains; in fact, some researchers say they solve all the problems a blockchain does and then some. Ironically, though, helping end-to-end verifiability go mainstream might end up being blockchains' greatest contribution to election security. After all, the word "blockchain" draws investor cash even to companies whose connection to the technology is, speaking generously, tenuous. And even skeptics acknowledge blockchains' relevance to voting; despite their questionable utility for security, similar procedures can enhance voting systems' efficiency or reliability.

This is why most blockchain election providers partially centralize the management of voter identities. Their systems are designed to query a consortium of several different identity databases such as government-issued

IDs and fingerprints collected during registration to match the voter with a name from government voter rolls. A quorum of these identity authorities can also revoke lost or stolen voting keys. Similarly, the companies partially centralize the validation process to guard against malicious influence: Instead of allowing anyone to become a validator, the government or party organizing the election designates a consortium of universities, nongovernmental organizations and such whose consensus determines what makes it onto the blockchain.

The challenge, in this study, will demonstrate that it is not just cryptography but engineering, building a scalable and reliable service in order to restore trust in the election process while US Government continues to function even under constant cyber-attack. And, as well, blockchain voting would require more than simply replacing Bitcoin transactions with votes. Which means that Instead, users generate public “addresses,” which act like deposit-only account numbers for receiving money, along with secret digital “keys” that are needed to transfer money out of the corresponding accounts.

To achieve secure cryptographic functionality such as encryption, signature or authentication, one must rely on computational problems that are presumed to be hard for the adversary. Since the study proposes a secure mobile device using a voting app (help election management bodies introduce online voting with levels of security and safety that are superior to in-person, paper-based elections, with restored trust, increased access, voter verifiability, and better public transparency.) on a secure Android/iOS mobile device using OKL4 with PitBull. Vulnerabilities in the Native OS Kernel can be exploited by applications. Native operation, like any other, begins with a hardware platform or SoC. Virtualization guarantees separation and provides the software foundation for security. Secure Boot establishes a root of trust and ensures software and hardware integrity. Selection of the correct SoC provides the hardware foundation for security. Virtualization is a critical technology that is part of the embedded security solution.



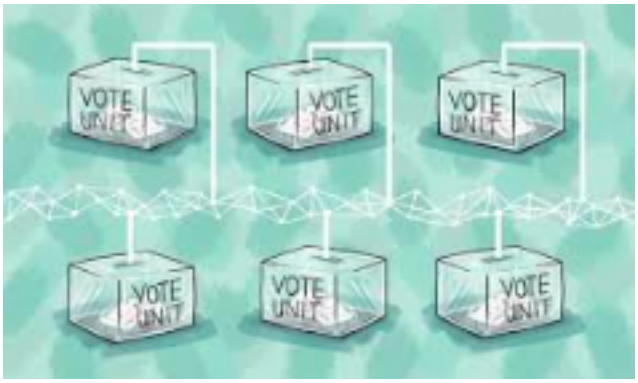
Another goal of the study is to ensure the key strength matches the sensitivity of the data. The length of the key, algorithm used, and the randomness of the key material are the main factors to consider in this area:

- Distribution - A key must be associated with a particular user, system, application or policy. The association will determine the requirements to secure the key, and the method used to secure it while in transit. The ability to differentiate access between the administrator creating the key and the person using it is vital.
- Storing - Organizations because using the same key over a long duration of time increases the chances of a compromise.
- Revocation - Every organization needs the ability to revoke, destroy or take keys offline. Backup copies of cryptographic keys should be kept in a storage mechanism that is at least as secure as the original store.
- Consistency policy enforcement requires the ability to provision and de-provision cryptographic resources, automate client provisioning, and create multi-tenant, tiered security administrator access levels.
- First, determine how many keys can be generated, and where they are stored. Continue to update variables in the system, such as back-up networks and users. Next, establish a policy for key usage, defining application and device access levels and to what extent they can perform.
- Lastly, secure, automated and unified logging and reporting are absolutely crucial to maintain requisite risk and compliance posture. Key ownership must also be clearly defined, and all modifications recorded and securely stored in order to provide an authentic and trusted audit trail of key state changes.
- Have the option of storing their keys within hardware or software.
- Rotation - Each key should be designated a lifespan with the ability to change that key on demand. Limit the amount of data encrypted with a single key

The secure mobile app voting will provide a constant vigilant viewpoint in developing new security procedures and cryptographic protocols to mitigate the threats from an ever-changing network environment. At every stage of development, from idea conception, to design, to prototyping, to delivery and

implementation, to maintenance, security is a fundamental requisite. Despite the measures, the authors of this study have taken the initiative to maximize security, while recognizing the possibility of vulnerabilities and have established policies and procedures that prepare the DoD and its organizations to both prevent and proactively detect signs of intrusion and malicious activity. The study addresses the facts that trust in elections and voter confidence are waning to historic lows and that elections remain a mission-critical, complex, highly regulated, and scrutinized process.

Conclusion. To realize this grand vision, it is impossible to envision the future of democracy where digital elections online are not the global standard. Unfortunately, there exists a wide gap between this inevitability and the progress actually being made towards secure online voting. In fact, most recent efforts, technically and legislatively, have been focused on in-person, paper-based voting. However, moving backwards towards in-person, paper-based elections ignore an increasingly demanding, mobile, and connected society with a need for broader access, public transparency, and individual verifiability. As opposed to working collaboratively to find solutions to these problems, many are trying pull the industry back into the past. Society no longer rides horse and buggy despite the risk of auto accidents introduced by cars; the mere existence of risk should not preclude technological progress. The studies' bottom line is to establish blockchains' potential to improve election integrity.



Consequently, in an effort to modernize the democratic process, the proposed study introduces a blueprint for secure online voting. The objective of this study is to assist and satisfy secure mobile apps whereby election management bodies can introduce online voting tools with levels of security and safety that are

superior to in-person, paper-based elections, with restored trust, increased access, voter verifiability, and better public transparency.

If election officials and election system vendors ignore these study recommendations, we expect that deficient, unverifiable Internet voting systems will be widely used within eight years. Vendors will claim to have solved the security problems, and eager officials will believe these claims. Elections may be altered with no public awareness. If election officials manage to find evidence left by a careless attacker after altering an election, the damage will have already been done.

References.

1. T. Abdoul et al. “AADL Execution Semantics Transformation for Formal Verification”. In: *13th IEEE Inter- national Conference on Engineering of Complex Computer Systems (ICECCS 2008)*. Mar. 2008, pp. 263–268. DOI: 10.1109/ICECCS.2008.24.
2. Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.
3. *Accessible Voting Technology Initiative*. URL: http://elections.itif.org/ (visited on 07/01/2015).
4. *Accessible Voting Technology Initiative: Election Design Guidelines*. URL: http://elections.itif.org/resources/guidelines/ (visited on 07/01/2015).
5. Claudia Z. Acemyan et al. “Usability of Voter Verifiable, End-to-end Voting Systems: Baseline Data for He- lios, Prêt à Voter, and Scantegrity II”. In: *The USENIX Journal of Election Technology and Systems* (2014), p. 26.
6. *ACSL: ANSI ISO C Specification Language*. URL: http://www.frama-c.com/download/acsl.pdf (visited on 07/01/2015).
7. *ADA Standards for Accessible Design*. URL: http://www.ada.gov/2010ADAstandards_index.htm (visited on 07/01/2015).
8. Ben Adida. “Helios: Web-based Open-Audit Voting”. In: *USENIX Security*. 2008. URL: https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf (visited on 07/01/2015).
9. Ben Adida et al. “Electing a University President using Open-Audit Voting: Analysis of real-world use of Helios”. In: *USENIX EVT/WOTE*. 2009. URL: https://www.usenix.org/legacy/event/evtwote09/tech/full_papers/adida-helios.pdf (visited on 07/01/2015).
10. *Alloy: A Language & Tool for Relational Models*. URL: http://alloy.mit.edu/ (visited on 07/01/2015).
11. *American Fuzzy Lop*. URL: http://lcamtuf.coredump.cx/afl/ (visited on 07/01/2015).

12. Jonathan Ben-Nun et al. “A new implementation of a dual (paper and cryptographic) voting system”. In: *Electronic Voting*. 2012, pp. 315–329.
13. Josh Benaloh and Michael de Mare. *Efficient Broadcast Time-Stamping*. Tech. rep. Jan. 1991. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=68476.
14. Cedric Beust and Hani Suleiman. *Next Generation Java Testing*. Addison-Wesley, 2007.
15. David Bismark et al. “Experiences Gained from the first Prêt à Voter Implementation”. In: *First International Workshop on Requirements Engineering for e-Voting Systems (RE-VOTE)*. IEEE. Aug. 2009, pp. 19–28. DOI: 10.1109/RE-VOTE.2009.5.
16. *BitBucket*. URL: http://www.bitbucket.org/ (visited on 07/01/2015).
17. Bruno Blanchet. “Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif”. English. In: *Foundations of Security Analysis and Design VII*. Ed. by Alessandro Aldini, Javier Lopez, and Fabio Martinelli. Vol. 8604. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 54–87. ISBN: 978-3-319-10081-4. DOI: 10.1007/978-3-319-10082-1_3.
18. Günter Böckle et al. “Calculating ROI for software product lines”. In: *Software* 21.3 (2004), pp. 23–31.
19. Dan Boneh, Amit Sahai, and Brent Waters. “Functional encryption: Definitions and challenges”. In: *Theory of Cryptography*. Springer, 2011, pp. 253–273.
20. Jan Bosch. “Maturity and evolution in software product lines: Approaches, artefacts and organization”. In: *Software Product Lines*. Springer, 2002, pp. 257–271.
21. Lionel C. Briand et al. “Exploring the relationships between design measures and software quality in object- oriented systems”. In: *Journal of systems and software* 51.3 (2000), pp. 245–273.
22. Philippe Bulens, Damien Giry, and Olivier Pereira. “Running mixnet-based elections with Helios”. In: *USENIX EVT/WOTE*. 2011. URL: http://www.usenix.org/events/evtwote11/tech/final_files/Bulens. pdf (visited on 07/01/2015).
23. Craig Burton et al. “Using Prêt à Voter in Victorian State elections”. In: *USENIX EVT/WOTE*. 2012. URL: https://www.usenix.org/system/files/conference/evtwote12/evtwote12-final9_0.pdf (visited on 07/01/2015).
24. Michael D. Byrne, Kristen K. Greene, and Sarah P. Everett. “Usability of voting systems: Baseline data for paper, punch cards, and lever machines”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2007, pp. 171–180.
25. *C0: Specification and Verification in Introductory Computer Science*. URL: http://c0.typesafety.net/ (visited on 07/01/2015).
26. Richard Carback et al. “Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Govern- mental Election with Ballot

Privacy". In: *USENIX Security*. 2010. URL: https://www.usenix.org/legacy/events/sec10/tech/full_papers/Carback.pdf (visited on 07/01/2015).
27. Carter Center. *Internet Voting Pilot: Norway's 2013 Parliamentary Elections*. Mar. 2014. URL: http://www. cartercenter . org / resources / pdfs / peace / democracy / Carter - Center - Norway - 2013 - study - mission-report2.pdf (visited on 07/01/2015).
28. *Center for Advanced Software Analysis: Software Tools*. URL: http://casa.au.dk/software-tools/ (visited on 07/01/2015).
29. Scott Chacon. *Pro Git*. Apress, 2014. ISBN: 978-1484200773.
30. David Chaum, Claude Crépeau, and Ivan Damgard. "Multiparty unconditionally secure protocols". In: *Pro- ceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. ACM. 1988, pp. 11–19.
31. Alex Delis et al. *Pressing the Button for European Elections 2014: Public attitudes towards Verifiable E-Voting In Greece*. June 2014. URL: https://drive.google.com/file/d/0BmtbRwyPn_SdnpMRzBKcEZWUm8/view?usp=sharing (visited on 07/01/2015).
32. Jared DeMott. "Revolutionizing the Field of Grey-box Attack Surface Testing with Evolutionary Fuzzing". In: *BlackHat and DefCon*. 2007.
33. David L. Detlefs et al. *SRC Research Report 159: Extended Static Checking*. Compaq Systems Research Cen- ter, Dec. 1998.
34. *DO-178B Software Considerations in Airborne Systems and Equipment Certification*. URL:http://www.rtca.org/store_product.asp?prodid=581 (visited on 07/01/2015).
35. *DO-178C Software Considerations in Airborne Systems and Equipment Certification*. URL: http://www.rtca.org/store_product.asp?prodid=803 (visited on 07/01/2015).
36. *Docker*. URL: http://www.docker.com/ (visited on 07/01/2015). Stefan Popoveniuc and Ben Hosp. "An Introduction to PunchScan". English. In: *Towards Trustworthy Elec- tions*. Ed. by David Chaum et al. Vol. 6000. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 242–259. ISBN: 978-3-642-12979-7. DOI: 10.1007/978-3-642-12980-3_15.
37. *Puppet Labs*. URL: http://www.puppetlabs.com/ (visited on 07/01/2015).
QuickCheck: Automatic testing of Haskell programs. URL: https://hackage.haskell.org/package/
38. QuickCheck (visited on 07/01/2015).
RAISE—Rigorous Approach to Industrial Software Engineering. URL: http://spd-web.terma.com/Projects/RAISE/ (visited on 07/01/2015).
39. *Redmine*. URL: http://www.redmine.org/ (visited on 07/01/2015)

40. *ReSharper*. URL: https://www.jetbrains.com/resharper/ (visited on 07/01/2015).
41. Ronald L. Rivest and John P. Wack. *On the notion of "software independence" in voting systems*. Prepared for the TGDC, and posted by NIST at the given URL. July 2006. URL: http://vote.nist.gov/SI-in-voting.pdf.
42. Noel H. Runyan. *Improving access to voting: A report on the technology for accessible voting systems*. 2007. URL: http://www.demos.org/publication/improving-access-voting-report-technology- accessible-voting-systems (visited on 07/01/2015).
43. *EasyCrypt: Computer-Aided Cryptographic Proofs*. URL: http://www.easycrypt.info/ (visited on 07/01/2015). *The Haskell Cabal: Common Architecture for Building Applications and Libraries*. URL: https : / / www . haskell.org/cabal/ (visited on 07/01/2015).
44. *The Haskell Lightweight Virtual Machine (HaLVM)*. URL: https://galois.com/project/halvm/ (visited on 07/01/2015).
45. Ken Thompson. "Reflections on Trusting Trust". In: *Communications of the ACM* 27.8 (Aug. 1984), pp. 761– 763. ISSN: 0001-0782. DOI: 10.1145/358198.358210.
46. Nikolai Tillmann and Jonathan de Halleux. "Pex—White Box Test Generation for .NET". In: *Proc. of Tests and Proofs (TAP'08)*. Vol. 4966. Lecture Notes in Computer Science. Prato, Italy: Springer-Verlag, Apr. 2008, pp. 134–153. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=81193 (visited on 07/01/2015).
47. James E Tomayko. "A comparison of pair programming to inspections for software defect reduction". In: *Computer Science Education* 12.3 (2002), pp. 213–222.
48. *Eiffel Inspector*. URL: https://docs.eiffel.com/book/eiffelstudio/eiffel-inspector (visited on 07/01/2015).
49. Aleks Essex et al. "Punchscan in practice: an E2E election case study". In: *Proceedings of Workshop on Trust- worthy Elections*. 2007. *Trac*. URL: http://trac.edgewall.org/ (visited on 07/01/2015). Georgios Tsoukalas et al. "From Helios to Zeus". In: *USENIX Journal of Election Technology and Systems* 1.1
50. (Aug. 2013). URL: https://www.usenix.org/jets/issues/0101/tsoukalas (visited on 07/01/2015). *United States Census Bureau*. URL: http://www.census.gov/ (visited on 07/01/2015).
51. *United States Election Assistance Commission: Accredited Test Laboratories*. URL: http://www.eac.gov/ testing_and_certification/accredited_test_laboratories.aspx (visited on 07/01/2015).
52. *UPPAAL*. URL: http://www.uppaal.org/ (visited on 07/01/2015). U.S. Election Assistance Commission. *UOCAVA Pilot Program Testing Requirements—August 25, 2010*. Aug.

53. 2010. URL: https://www.fvap.gov/uploads/FVAP/VSTL_AppendixB.pdf (visited on 07/01/2015).
54. *Verasco*. URL: http://verasco.imag.fr/ (visited on 07/01/2015).
55. *EMMA: A free Java code coverage tool*. URL: http://emma.sourceforge.net/ (visited on 07/01/2015).
56. Michael D. Ernst et al. “The Daikon system for dynamic detection of likely invariants”. In: *Science of Com- puter Programming* 69.1–3 (2007), pp. 35–45. DOI: 10.1016/j.scico.2007.01.015.
57. *Verified Software Toolchain*. URL: http://vst.cs.princeton.edu/ (visited on 07/01/2015).
58. *Verifying Multi-threaded Software with Spin*. URL: http://spinroot.com/ (visited on 07/01/2015).
59. Kim Waldén and Jean-Marc Nerson. *Seamless object-oriented software architecture: Analysis and design of reliable systems*. New York: Prentice Hall, 1995. ISBN: 0130313033.
60. *Web Accessibility Evaluation Tool*. URL: http://wave.webaim.org/ (visited on 07/01/2015). *Web Accessibility Initiative*. URL: http://www.w3.org/WAI/ (visited on 07/01/2015).
61. David A. Wheeler. “Fully Countering Trusting Trust through Diverse Double Compilation”. PhD thesis. George Mason University, 2009. URL: http://www.dwheeler.com/trusting-trust/ (visited on 07/01/2015).
62. *Xen Project*. URL: http://www.xenproject.org/ (visited on 07/01/2015). Xuejun Yang et al. “Finding and understanding bugs in C compilers”. In: *ACM SIGPLAN Notices*. Vol. 46. 6. ACM. 2011, pp. 283–294.
63. Shin Yoo and Mark Harman. “Regression testing minimization, selection and prioritization: a survey”. In: *Software Testing, Verification and Reliability* 22.2 (2012), pp. 67–120.
64. *YouTrack*. URL: https://www.jetbrains.com/youtrack/ (visited on 07/01/2015).
65. Filip Zagórski et al. “Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System”. English. In: *Applied Cryptography and Network Security*. Ed. by Michael Jacobson et al. Vol. 7954. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 441–457. ISBN: 978-3-642-38979-5. DOI: 10.1007/978-3-642-38980-1_28.
66. Daniel M. Zimmerman and Rinkesh Nagmoti. “JMLUnit: The Next Generation”. In: *International Conference on Formal Verification of Object-Oriented Software (FoVeOOS 2010)*. Paris, France, June 2010.
67. *Guide for Applying the Risk Management Framework to Federal Information Systems*. URL: http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf (visited on 07/01/2015).
68. Stuart Haber and W. Scott Stornetta. “How to Time-Stamp a Digital Document”. In: *Advances in Cryptology— CRYPT0’ 90*. Ed. by Alfred J. Menezes and Scott A. Vanstone. Vol. 537. Lecture Notes in

Computer Science. Springer Berlin Heidelberg, 1991, pp. 437–455. ISBN: 978-3-540-54508-8. DOI: 10.1007/3-540-38424- 3_32.
69. Øystein Haugen, Andrzej Wa ̨sowski, and Krzysztof Czarnecki. “CVL: Common Variability Language”. In: *Proceedings of the 16th International Software Product Line Conference-Volume 2*. ACM. 2012, pp. 266–267.
70. James Herbsleb et al. “Software quality and the capability maturity model”. In: *Communications of the ACM* 40.6 (1997), pp. 30–40.
71. *HOL4*. URL: http://hol.sourceforge.net.
72. Gerard J. Holzmann. *UNO: Static source code checking for user-defined properties*. 2002. URL: http://www. spinroot.com/uno/uno_long.pdf (visited on 07/01/2015).
73. *How to Vote: Wombat Voting System*. URL: http://www.wombat-voting.com/how-to-vote (visited on 07/01/2015).
74. Engelbert Hubbers, Bart Jacobs, and Wolter Pieters. “RIES - Internet Voting in Action”. In: *29th International Computer Software and Applications Conference (COMPSAC 2005)*. IEEE. 2005, pp. 417–424.
75. Engelbert Hubbers et al. “Description and analysis of the RIES internet voting system”. In: *Report of the Eidhoven Institute for the Protection of Systems and Information*. Faculty of Mathematics and Computer Science Eindhoven University of Technology, June 2008.
76. *HUnit-Plus: A test framework building on HUnit.* URL: https://hackage.haskell.org/package/HUnit -Plus (visited on 07/01/2015).
77. *IEC 62304 - Medical Device Software - Software life cycle processes*. URL: http://www.iso.org/iso/ catalogue_detail.htm?csnumber=38421 (visited on 07/01/2015).
78. *IEEE 1622-2011 - IEEE Standard for Electronic Distribution of Blank Ballots for Voting Systems*. URL: http: //ieeexplore.ieee.org/servlet/opac?punumber=6130554 (visited on 07/01/2015).
79. Laura Inozemtseva and Reid Holmes. “Coverage is not strongly correlated with test suite effectiveness”. In: *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. ACM. 2014, pp. 435–445.