

VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS  
INFORMATIKOS KATEDRA

**Blokų grandinės pagrindu veikiančios  
elektroninio balsavimo sistemos saugumo analizė**  
**Security Analysis of a Blockchain-Based E-voting System**

Magistro baigiamasis darbas

Atliko:	Miglė Babickaitė	(parašas)
Darbo vadovas:	Dr. Linas Bukauskas	(parašas)
Konsultantas:	Ch.mo prof. Simon P. Romano	(parašas)
Recenzentas:	Tomas G. Lipnevičius	(parašas)

Vilnius – 2024

## Santrauka

Elektroninis nuotolinis balsavimas yra sudėtinga ir intriguojanti kompiuterių saugumo inžinerijos užduotis. Nuo bet kokios kitos internetinės veiklos ji skiriasi savo įtaka demokratijai. Nepavykus užtikrinti tokios sistemos saugumo, didelio masto pasekmės gali anuliuoti rinkimus, kurių pakartotinis skelbimas yra ne tik brangus, bet ir žalojantis visuomenės pasitikėjimą. Dėl rinkimų rezultatų svarbos, aukštos kvalifikacijos užpuolikai gali atakuoti žemo kompiuterinio raštingumo vartotojus. Lietuva elektroninio balsavimo idėją puoselėja nuo 2016 m., tačiau konstitucinių reikalavimų neatitiko jokie realūs pasiūlymai. Dėl pastaraisiais metais drastiškai išaugusio nuotolinio aktyvumo, Lietuvos vyriausioji rinkimų komisija užsakė dar vieną nuotolinio elektroninio balsavimo galimybių studiją. Atsižvelgiant į šį kontekstą, šis darbas apibrėžia kritiškiausius elektroninės balsavimo sistemos taškus ir jos saugumo reikalavimų įgyvendinamumą. Šiam tikslui pasiekti atliekama Neapolio universitete sukurtos blokų grandinės ir susietais žiediniais parašais pagrįstos elektroninio balsavimo sistemos *Chirotonia* tikimybinė saugumo analizė. Analizė atliekama remiantis Bajeso atakų grafomis, kurios automatiškai sugeneruojamos iš turimų žinių.

**Raktiniai žodžiai:** Elektroninis balsavimas, saugumo tyrimas, blokų grandinė, Bajeso atakų grafos, ontologija

## Summary

Electronic remote voting is a challenging and intriguing computer security engineering task. The crucial impact that compromised elections have on democracy and voters' trust differs from any other internet activity. Due to the importance of the election outcome, highly skilled adversaries might attack users with low computer literacy. Lithuania has been grooming the idea of e-voting since 2016, however, real proposals did not satisfy the constitutional requirements and have been rejected. Due to the drastically increased remote activity in recent years, another remote electronic voting feasibility study was commissioned by the Lithuanian Central Election Commission. In light of this context, this thesis defines the critical aspects of an electronic voting system and the feasibility of its security requirements. As a methodology, the security analysis of the blockchain and linkable ring signature-based electronic voting framework *Chirotonia* developed at the University of Naples Federico II is performed. The analysis is based on Bayesian attack graphs which are automatically generated based on the obtained knowledge.

**Keywords:** Electronic Voting, Security Analysis, Blockchain, Linkable Ring Signatures, Ontology, Bayesian Attack Graph

## Contents

1	Introduction .....	4
2	Related Work .....	6
2.1	Electronic Voting .....	6
2.1.1	Electronic vs. Paper-Based Voting .....	6
2.1.2	Secure Voting Requirements.....	7
2.1.3	Requirement Implementation .....	11
2.2	Risk Assessment .....	12
3	Methodology .....	17
3.1	Ontology Development.....	18
3.2	Knowledge Bases.....	20
3.3	Common Vulnerability Scoring System .....	20
3.4	Bayesian Attack Graphs.....	20
3.5	Automatic Graph Generation.....	22
3.6	Risk Assessment .....	22
4	Ontological Knowledge Acquisition .....	23
4.1	Regulation Analysis.....	23
4.2	System Analysis: <i>Chirotonia</i> .....	24
4.2.1	Assets .....	24
4.2.2	Locations .....	26
4.2.3	Phases.....	27
4.2.4	Election Compromises .....	31
4.2.5	Weaknesses .....	31
4.2.6	Mitigations .....	31
5	Risk Assessment .....	33
5.1	Ballot Altering .....	33
5.2	Voter Privacy Breach.....	34
5.3	Voter Disenfranchisement .....	35
5.4	Multiple Ballots .....	36
5.5	Non Eligible Voters.....	36
6	Discussion .....	38
7	Conclusion and Future Work .....	39
	References .....	40

# 1 Introduction

According to the in-depth analysis performed by the Scientific Foresight Unit in the European Parliament [Eur18], digital development has potential to reduce the democratic deficit in the European Union. However, even though participants gain added personal value, different types of e-participative projects suffer from a lack of direct or indirect political impact. For example, *e-consultations* are meant to allow participatory decision-making but often turn out to inform citizens about decisions that have already been made. One of the prominent types of e-democracy is the *e-voting*. EU's democratic deficit is strongly manifested in continuously decreasing electoral participation. The analysis observes that the convenience aspect is not the only influence on whether a citizen votes, and the Internet voting is not a quick technological fix to the lost political interest and satisfaction. The rise of blockchain technology gave a new hope for the Blockchain-enabled voting [Bou16]. Initially meant to track cryptocurrency transactions, the open and untamperable blockchain ledger is considered to be able to empower decentralised and direct democracy.

Following the trends and recommendations in the European Union, the idea of electronic voting in Lithuania was proposed in 2006. Since then, multiple projects have been drawn up and rejected by the Parliament but the relevance of the topic has not faded. With a good amount of Lithuanians living abroad, the possibility of electronic remote voting could theoretically increase voter turnout substantially. On the other hand, security requirements for electronic voting are considerably more sensitive than any other electronic operation. Having in mind the geopolitical situation in the region and past attempts to sabotage national elections [Cla14], guaranteeing the security of the overall e-voting protocol and its implementation is a complex and dangerous task. The common example that pro-e-voting lobbyists provide to support their argument is the supposed success of e-voting in Estonia [CM16]. However, an extensive security analysis [SFD<sup>+</sup>14] has quite harshly concluded that the Estonian system is not suitable for anonymous and secure electronic voting and should be discontinued immediately.

The COVID-19 pandemic expanded our digital presence greatly. In continuation of this trend, in 2021, yet another online voting feasibility study was commissioned by the Lithuanian Central Election Commission (CEC) [rkom21]. The ordered analysis must identify the online voting system goals, assess the assurance of system security and reliability, envision the lifecycle of the system development and testing, and evaluate organisational procedures. In the context of national electronic voting, e-democracy starts before the election event. Every phase of the decision-making must be transparent and regulated. The choice of the adopted e-voting system must be informed and democratic. Legal regulations must be constantly refined and systems' legal compliance verified and audited. Early in 2003, Walker et al. [WHR<sup>+</sup>03] noticed the lack of common understanding between the policy decision-makers and scientific decision supporters due to missing definitions of different dimensions of uncertainty. Electronic voting is a vast process with many possible actors of different capabilities and goals. The gathered knowledge and sensitively calculated uncertainty are crucial to risk assessment of any system considered for the carrier of e-democracy.

As a starting point for a more extensive feasibility study, this thesis analyses the knowledge-based electronic voting system verification and decisions that such verification affords. Community maintained Common Vulnerability Enumeration (CVE), Common Weakness Enumeration (CWE) and Common Attack Pattern Enumeration and Classification (CAPEC) bases provide the knowledge for the system analysis. To connect the provided vast enumeration resources and align them with the system requirements, an ontology is proposed. Ontologies are a powerful tool for any task that deals with systemized knowledge. In the ontology-based Cloud threat modeling performed in the [MVT<sup>+</sup>19], a developed ontology is mapped onto Design Structure Matrices (DSM) to visualise interactions between requirements, services and vulnerabilities. Similarly, this work maps the developed ontology onto Bayesian Attack Graphs (BAG). These graphs provide visualisations and quantified conclusions about the most vulnerable system components, the most probable attack paths and required mitigation strategies to control the risks.

To demonstrate the proposed verification methodology, a case security analysis is performed on a blockchain-based electronic voting system *Chirotonia*. The analysis demonstrates that the system implementation fails to deliver the promises made by the blockchain-based architecture.

The contributions of this thesis are as follows.

1. The development of an ontology based framework to identify weak system components and design choices.
2. The security evaluation of a blockchain-based electronic voting system.

## 2 Related Work

To understand the broader context, two types of related work are analysed: (i) electronic voting requirements and their implementation are analysed in the Section 2.1; (ii) research on risk assessment and especially, legally-bound risk assessment, methods is analysed in the Section 2.2.

### 2.1 Electronic Voting

#### 2.1.1 Electronic vs. Paper-Based Voting

Electronic remote voting is a challenging and intriguing computer security engineering task. Even though almost every aspect of our daily lives has been automated, only a few countries vote electronically. There are legitimate reasons why most of our democratic voting is performed via paper-based physical protocols.

- **Security** As authors of [PSN<sup>+</sup>21] pointed out, online voting systems require different security properties than online banking or cryptocurrency transactions. Online shopping and banking systems are more failure-tolerant as banks and insurers absorb the risk. There is no insurance against the failure of democratic voting. Additionally, a threat profile for online voting is very different from that of other online systems. In an online voting context, a highly skilled attacker might attack a technically unsophisticated voter. Moreover, attacks on electronic voting systems are highly scalable — a remote programmer changing a line of code could in principle change millions of electronic ballots in milliseconds, whereas changing millions of paper ballots requires physical access and one-by-one handling.
- **Decreased Transparency** Electronic voting infrastructure includes many hardware and software components that are not conceivable to most of the general public. Tallying and vote counting is performed electronically and is as much trustworthy as every single part of the supply chain, starting with the installation of the operating system to the vote-handling servers. As the security analysis of the Estonian e-voting system noted, an attacker who strikes early enough can introduce malicious code into the counting server by using a chain of infections that parallels the configuration process [SFD<sup>+</sup>14]. In that case, malicious malware can be installed into servers early on, and the security of the voting protocol loses its relevance immediately (unless it assumes by design that no part of the system is trustworthy).
- **Authentication Complexity** Having to verify the user's identity while also breaking ties between the identity and the ballot the user cast is a unique challenge to online voting.

Having the risks in mind, scientific e-voting discussions have been developing for over forty years without the loss of relevance. The theoretical advantages offered by electronic voting entice researchers and fuel new creative proposals.

- **Accuracy** There is no doubt that ballot processing accuracy and speed could be drastically improved with electronic voting. Human error can be greatly reduced and system auditing simplified.
- **Accessibility** In theory, electronic voting could provide easier voting access to people, however varied technological sophistication between different social groups must be considered when designing the system. As the research performed in [PJS21] concludes, the availability of e-voting does have an influence on turnout, but this influence holds for specific groups of citizens only.
- **Verifiability** The third advantage of electronic voting is vote verifiability. State-of-the-art cryptographic primitives can provide the ability to verify the published election result. In theory, a voter is able to verify that their vote was processed correctly and that the published election result corresponds with the set of published votes. Individuals usually cannot verify, without trusting some authority, that their paper ballot was counted correctly [Wil22]. In 2004, D. Chaum proposed a solution that allows each voter to verify with visual cryptography that their paper ballots are cast appropriately and accurately tallied [Cha04]. After the voter selects their candidates, a voting machine prints out a specially formatted version of the ballot on two transparencies. When the layers are stacked, they show the human-readable vote. However, each transparency is encrypted with a form of visual cryptography so that it alone does not reveal any information unless decrypted. The introduction of electronic voting allows for many cryptographic solutions to system verifiability.

### 2.1.2 Secure Voting Requirements

Requirements for secure voting depend deeply on the political, sociological, ideological, legal and regulatory framework in which the voting occurs. Even within Europe, the regulations for e-voting vary widely. While some countries, like Estonia and Switzerland, are eager to vote electronically and define detailed e-voting regulations, in other countries e-voting is explicitly forbidden. In Germany, a 2009 ruling ruled voting machines and electronic voting unconstitutional in their current form. Although the German Constitutional Court did not forbid electronic voting from ever being utilised in elections in the future, it concluded that electronic voting systems can only meet the constitutional requirements if it is possible for voters or observers without specialised knowledge to verify the results of the election [Ins09]. In the Netherlands, a court ruling in 2007 concluded that elections should be conducted with pen and paper due to integrity issues with voting machines [Ins07].

In 2017, the European Union published recommendations for e-voting standards [Cou17]. On a high level, the recommendations include:

- **Universal suffrage** All eligible voters can participate in the election process. This means that every eligible person should be able to register for participation in elections without discrimination.



- **Equal suffrage** Each vote should be counted equally. Additionally, ballots should not give a preference to any candidate or party.
- **Secret suffrage** No actor in the voting process, such as election officials or observers, should be able to trace the voter's identity given their ballot. During all phases of the election process, the secrecy of the vote should be protected, and the voter should not be able to prove that they voted in a certain way.
- **Transparency** Any observer, to the extent permitted by law, shall be enabled to observe and comment on the e-elections, including the compilation of the results.
- **Accountability** Before an e-voting system is introduced and at appropriate intervals thereafter, and in particular after any significant changes are made to the system, an independent and competent body shall evaluate the compliance of the e-voting system and of any information and communication technology (ICT) component with the technical requirements. This may take the form of formal certification or other appropriate control.
- **Reliability** The system should provide a reliable and correct voting result.

In Lithuania, the electronic voting regulations are drafted in the the Internet voting draft law proposed in 2018 by the Ministry of Justice of the Republic of Lithuania [Min18]. The list below extracts the *electronic voting process* requirements. The requirements for governance, internal, and other administrative processes are not analysed in this work as the performed analysis in its scope and complexity does not compare to the national election that the law defines.

**Last Vote Precedence** Article 3 of the draft law requires a voter to be able to cast as many votes as they want. Only the last vote should be included in the final tally. Essentially, this requirement endangers voter anonymity as for the system to rewrite a voter's ballot, the ballots must be linkable together up until the final tally stage. Ballot linkability can provide an attacker with information that helps identify the voter. Truly anonymous systems break the link between the ballot and its owner before the vote submission, assuming that the submission channel is unconditionally anonymous [HMM<sup>+</sup>23].

**Accessibility** Article 6 of the draft law requires voters with minimal computer literacy to be able to vote. Voters must be able to vote from end Internet devices that meet the technical properties defined by the system administrator.

**Authentication** Article 7 of the draft law defines voter authentication requirements. Voters must be identified and verified using electronic identification methods that meet the high-security level defined in [iEur14]. The system must prevent the use of other parties' electronic identification.

**Eligibility** Article 7 details the eligibility requirements as well. Only the votes of eligible voters should be included in the final tally. Technologically, this means that eligible voters need to be authenticated, and registered prior to the voting phase. Voting authorities must have a list of all registered eligible voters' (anonymous) identifiers against which the identity information provided in the ballot is matched.

**Verifiability** Article 9 of the draft law defines verification requirements. After an election, it should be possible to verify that the final public voting result corresponds correctly to the private voting ballots. A voter can verify that:

- *cast-as-intended (CAI)*: their choice was correctly denoted on the ballot by the system.
- *recorded-as-cast (RAC)*: their ballot was recorded the way they cast it.
- *tallied-as-recorded (TAR)*: their ballot counts as received.

The draft law defines a ballot box as a private, technologically and physically secured component. This contradicts the verifiability requirement as a voter does not have access to the ballot box and cannot perform the verification procedures. Consequently, the verification can only be performed internally and the voter can only *ask* the system to verify the current state for them. This implies unconditional trust in the central voting system which is difficult to achieve.

**Privacy** Article 11 of the draft law details the privacy requirements. Only the voter themselves should know the contents of their voting ballot. No other party, not even the administrator of the system, should be able to uncover the voter's ballot. It is forbidden to collect material that could reveal the contents of ballots. The draft law defines some technical means to reach this goal:

- Electronic ballot must be encrypted in the voter's end device. No plain data can travel via the Internet.
- Electronic ballot box must hold only the last vote of the voter.
- The privacy of the encrypted ballots must be preserved until the election's final day. This implies that ballot privacy is not everlasting, i.e. a voter's ballot is not private after the election.
- Electronic double-envelope principle must be applied. The inner envelope holds the encrypted electronic ballot and the outer envelope holds the voter's identity information. Before the tallying phase, the inner envelope is irreversibly separated from the outer envelope and mixed with other inner envelopes. The ballots are decrypted only after all the envelopes have been mixed.
- No party can modify ballots, or add non-legitimate ballots to the ballot box.
- Before the election, the CEC creates the cryptographic key pair and is responsible for their storage until the election.

**Auditability** Article 14 of the draft law describes the desired system auditability. The system must be auditable by the system administrator and auditors. Audits should not violate ballot privacy. All voting actions and all access to the system must be logged in a separate database.

However, the draft law lacks important requirements that are crucial to a truly democratic election execution.

**Coercion-Resistance** Coercion-resistance requires that a voter cannot be influenced by another party during an election. This includes forced randomness (a voter is forced to vote randomly), forced abstention (a voter is forced not to vote), coerced vote (a voter is forced to vote in a predefined way), simulated vote (another party has impersonated a voter). The term first appeared in [JCJ05]. Usually, coercion-resistance also implies the requirement of **receipt-freeness** which states that a voter cannot prove how they voted after an election. This preserves the voter's privacy and enhances the coercion resistance, as there is no way that the voter can prove to the coercer how they actually voted. The concept first appeared in [BT94].

The draft law does not include the coercion-resistance requirement which is a huge shortcoming in a poor and politically uneducated country like Lithuania. The precedence of the last vote does not prevent coercion if at any point a voter (and by extension anyone who is supplied with the cryptographic material owned by the voter) can open their electronically cast ballot for verification purposes. Only in case the coerced electronic voter additionally casts a physical ballot can the voter's legitimate vote be unproven to a coercer. However, as the electronic vote is deleted immediately in such cases, the coercer would know that the voter changed their vote.

**Anonymity** Anonymity expands the requirement of privacy to conceal not only the contents of a ballot but also the act (or absence) of voting itself. Anonymity is very important in any democratic election. A link between a voter and their ballot, even if encrypted, can leak sensitive information that makes it possible to force a voter not to vote, verify the time when they voted, etc. A significant group of voters' security is breached if anonymity cannot be guaranteed. Moreover, voters should remain anonymous to legitimate election authorities as well, since otherwise gerrymandering<sup>1</sup> is made possible.

**Trust Distribution and Transparency** The draft law assumes unconditional trust in a closed central e-voting system. The system is managed by the system administrator and audits are performed by delegated auditors. The ballot box is private and the vote verification is performed internally. Voters can only *trust* the result that the system provides them with. In Article 15, the law defines two aspects of system transparency:

1. Dedicated observers and auditors can observe the system log to verify the correct flow of the procedure. This requires high technical knowledge and an understanding of the inner workings of the system design. Training independent observers becomes noticeably expensive.

---

<sup>1</sup>Gerrymandering is the political manipulation of electoral district boundaries with the intent to create undue advantage for a party, group, or socioeconomic class within the constituency.

2. The access to information about the system’s structure, design, and software is restricted. Firstly, this increases bug risk, as bugs in open-source software are found much faster. Moreover, this drastically reduces trust in the system and the amount of democracy it affords.

### 2.1.3 Requirement Implementation

Generally, remote e-voting systems include voters, authorities (registrars), tallies, a bulletin board, and potentially some other system-specific components such as auditors. The main role of authorities is to authenticate voters’ eligibility. This can be done by identifying the voter via some external mechanism and assigning them an anonymous token which, provided together with the filled ballot, proves their eligibility. Another way for the voters to show their eligibility is by signing ballots with a certified secret key. Usually, the same tools that prove voter eligibility prove vote uniqueness as well (Table 1).

All ballots are cast on the bulletin board. Usually, for the sake of transparency, verifiability and auditability, the bulletin board is public. Older proposals, such as the first version of Helios, model the public bulletin board as a trusted web server, however, a decentralised untamperable append-only public bulletin board is offered by the rise of blockchain technology. Some bulletin boards are designed over a permissioned blockchain which reduces the complexity and computational demand of the system but eliminates openness and verifiability. Some protocols like *sVote* [HLP<sup>+</sup>20] and SK95[SK95] substitute the public bulletin board with public proofs of correct protocol execution. This solution does not provide enough transparency to trust the system from a voter’s perspective.

To preserve voters’ privacy, all ballots come to the bulletin board encrypted. In order not to supply voters with vote receipts, voters do not encrypt their ballots themselves. Instead, they receive already encrypted ballots and non-transferable proofs of encryption contents. Voters select encrypted preferences and cast the filled-in ballot. In most of the systems, voters also sign their ballots thus forming a double-envelope structure: the outer envelope contains the voter’s identification and can be publicly verified (in the case of a public bulletin board), and the inner envelope contains the encrypted ballot. As discussed above, an alternative to the outer envelope is an anonymous token/credential included in the submitted input.

In order to mitigate possible coercion, some systems allow voters to submit multiple votes. In that case, only the last one is counted. Another solution for coercion-resistance allows deceiving the coercer. Voters can either cast fake ballots (that will be removed before the final tally) or provide coercers with a fake receipt indistinguishable from an honest one.

After the voting phase is over, in the traditional e-voting approach, tallying authorities verify voters’ identification, remove duplicate ballot submissions, shuffle ballots (e.g. with mix-nets), remove fake submissions, remove voters’ identification data (separate the voter’s identity from the ballot), decrypt and count the anonymised ballots. Alternatively, encrypted ballots can be added together via homomorphic encryption, and decryption can be performed on the election result only. Both of these methods preserve everlasting voter anonymity and ballot secrecy. However,

some systems, like *Chirotonia* [RAV<sup>+</sup>21], make ballots publicly open after the voting phase is over. This kind of scheme provides the public with a lot of information that might endanger voters' anonymity and confidentiality.

After the election result is announced, the amount of verifiability available to a voter differs among all reviewed proposals. Systems like Selene [RRI16] or Estonian [SFD<sup>+</sup>14], allow individual vote verifiability via a tracker. However, the final result can be universally verified via complex zero-knowledge proof verifications which does not increase trust in an average user. The Estonian system introduced a centralised component to perform complex universal verifiability on the votes' behalf. However, it is evident that the central component must be unconditionally trusted and secured. The correct execution of the protocol can be auditable via analysis of system logs or blockchain entries. This can only be done by dedicated, trained auditors and cannot be fully considered as *universal* verifiability.

It can be observed that electronic voting systems, like all software systems, tend towards decentralisation which eliminates the risk of attacks on a single point of failure. This is achieved by distributing authority and tallying roles between multiple agents. Usually a single honest agent is needed for the integrity of the system to be preserved. Additionally, as electronic voting becomes a more real possibility, modern research shifted slightly towards user accessibility.

## 2.2 Risk Assessment

In 2020, Taş et al. reviewed the challenges and opportunities of blockchain-based electronic voting in the [TT20]. They observe that blockchain technology contributes to ensuring confidentiality, data integrity and fault tolerance. The distributed and decentralised nature of such network mitigates one of the crucial threats to critical Internet system – the denial of service attacks. On the other hand, blockchain systems require software infrastructure like any other e-voting system, which are used to add or view the immutable data on the blockchain. These intermediate components and the open Internet itself are sensitive to many known and unknown software, hardware, supply chain, configuration vulnerabilities. As an example, a black-box security analysis of a blockchain-based e-voting system Voatz [SKW20] discovered that the untamperability promised by the blockchain is subverted by a vulnerable proxy web server which is capable of modifying voter's ballots. The voter does not have direct access to the blockchain and therefore the blockchain-promised transparency is blurred by the web server handing over and possibly modifying or simply forging verifiable receipts. Another observation made in the [TT20] and supported in the [PSN<sup>+</sup>21] is that blockchain architecture introduces *additional* problems for voting systems, namely vulnerabilities in smart contracts or 51% attacks. Knowledge on blockchain-specific attacks is scattered and not enumerated which makes it harder to verify potential systems.

Knowledge is crucial throughout the decision-making process. The knowledge must be gathered and organised, uncertainties defined and evaluated. Early in 2003, Walker et al. [WHR<sup>+</sup>03]

Table 1. Proposals of e-voting requirements' implementation in the literature.

ENC — ballot encryption; ZKP — zero-knowledge proofs; HM — homomorphic encryption; SIGN — signatures; COM — commitments; MPC — multiparty computation; LOG — logging; MN — mix-nets; TOKEN — token; TRACKER — tracker; S-RAND — randomness in encryption is secret to the voter; DECEIT — possibility for a voter to deceive a coercer; AUD — publicly open audit; BC — blockchain; LVP — last vote precedence; PHY — physical solution (e.g. postal mail, voting booth, etc.); CENT — central trusted authority.

Requirement	Techniques used in the research case									
	BT94	SK95	Helios	JCJ05	Selene	OVN	Voatz	Estonian	sVote	Chirotonia
Eligibility	TOKEN	-	TOKEN	TOKEN	SIGN	SIGN	TOKEN	SIGN	SIGN	SIGN
Privacy	ENC, ZKP, MPC, HM, PHY	ENC, ZKP	ENC, COM	ENC, ZKP	ENC, ZKP	MPC, ENC	ENC	ENC, HM	ENC, ZKP	ENC, MPC
Accessibility	-	-	CENT	-	TRACKER	-	CENT	CENT	-	-
Uniqueness	TOKEN	-	TOKEN	TOKEN	SIGN	SIGN	TOKEN	SIGN	SIGN	SIGN
Individual Verifiability	-	ZKP	LOG, AUD	-	TRACKER	BC	TOKEN	TRACKER	PHY	BC
Universal Verifiability	-	ZKP	LOG, AUD	-	ZKP	BC	-	CENT, ZKP	ZKP	BC
Auditability	-	-	LOG, AUD	-	-	BC	-	LOG, ZKP	-	BC
Coercion-resistance	PHY, S-RAND, HM	DECEIT, S-RAND	S-RAND	DECEIT	DECEIT	-	LVP	LVP	-	-
Anonymity	-	MN	MN, ZKP	TOKEN	-	TOKEN	TOKEN	MN	-	SIGN
Transparency	-	-	LOG, AUD	-	-	BC	-	-	-	BC

noticed the lack of common understanding of the different dimensions of uncertainty. This obstructs the communication and trust between the policy decision-makers and scientific decision supporters ultimately leading to worse policies. To solve this issue, the researchers provide harmonised terminology and a systematic uncertainty matrix for the model-based decision support activities. The uncertainty is defined as any *deviation from the unachievable ideal of completely deterministic knowledge of the relevant system*. It is considered to have three dimensions: the *location*, the *level* and the *nature* of uncertainty. A deterministic model can be transformed into a non-deterministic state in an uncertain environment. Verification of such a non-deterministic system becomes challenging due to the increased state space.

In an electronic voting context, the uncertainty lies in the adversarial capabilities and their probabilities. Due to the potential complexity of qualitative security models, minor changes might significantly impact their quantitative evaluation. Therefore, the verification task is composite: how to verify the legal compliance of a system in an uncertain election setting? The decision of whether the particular system is legally compliant is based on the system's representative model. As the complexity of the system enhances, model abstractions are introduced. It becomes harder to predict the individual components making the exhaustive test coverage crucial. Deviations caused by environmental changes have to be taken into account in the process of verification and validation to stamp the correctness of the system.

In 2007, a fuzzy logic-based threat modelling technique was proposed in the [SOO07]. Although not widely used for Internet voting systems, the fuzzy logic-based verification paves the way for uncertainty handling in classical threat modelling. In the proposed technique, the STRIDE model is used to determine the input variables passed to the fuzzy inference engine. The crisp numeric values indicating the severity of each threat (calculated via DREAD or any other risk assessment model) and the output threat level variable are fuzzified - using the predefined membership functions the variable membership values are calculated. Based on predefined fuzzy rules, the inference engine outputs the calculated defuzzified crisp threat level numeric value. The technique allows evaluation of the system security based on the level to which a particular threat is possible in the specific election setting.

Another well-known technique to validate the system under uncertainties is numerical simulation. The evaluation is done of a single aggregation of uncertainties. Multi-run simulation analysis is used, which requires several runs to enhance the coverage. However, this results in a high degree of confidence. One of the techniques is the Monte Carlo analysis [MS02]. In this case, the statistical properties of an uncertain system are determined. Here the values are chosen randomly from probabilistic models. This is in the future carried out in the repeated simulation runs. In [Neu16], Neumann proposes an e-voting scheme evaluation framework based on Monte-Carlo simulations capable of evaluating the electronic voting schemes in a manner readable by election officials. The evaluation process takes an Internet voting scheme, qualitative security models and the election setting (probability distributions of adversarial capabilities) as input and outputs the scheme's satisfaction degree. Monte-Carlo simulations randomly samples the adversarial capabilities according to the provided probability distributions. Then the probability that

an adversary might cause a certain impact on a specific security requirement in a specific scheme is calculated and multiplied by the normalised impact which yields a risk value. The satisfaction degree is the inverse of the largest risk value of all the impact levels calculated for the specific election setting.

An alternative to simulations for the evaluation of an uncertain environment is formally modelling of all possible states of all possible components of the system. However, as [CR20] conclude, formal methods are difficult to integrate into the flow of the design, and have inherent scalability issues due to the state-space explosion. Additionally, formal verification lacks the abstraction needed to reapply the verification process to multiple systems of different architectures and make informed decisions and is generally an expensive method.

One way to optimise the resources and effort required to verify and protect the system, is to assess its risks and prioritize the most critical threats. This includes estimating the risk exposure of vulnerable network nodes based on the knowledge of the threat likelihood and severity of its impacts. For such analysis to be adequate, the dependencies between vulnerabilities must be considered. Attack graphs [SH]<sup>+</sup>02 represent prior knowledge about vulnerabilities and network connectivity. They provide unified, visualised and perceivable way for the decision-makers and decision-supporters to reason about threats, their risks, and possible countermeasures which is known as *static analysis*. Additionally, attack graphs can be used to *dynamically* profile the attacker's paths to determine the possible future in an ongoing attack. Both, static and dynamic analysis have inherent probabilistic characteristics given the uncertainty about the attackers' ability to exploit known vulnerabilities. In this sense, Bayesian networks provide an appropriate framework to model attack graphs since they depict causal relationships between random variables in a compact way.

The use of Bayesian networks for attack graph modelling was first introduced in [LM05] in 2005. The authors used Bayesian networks to calculate the probability that an attacker can reach a security state given prior knowledge of the state it had reached. In [PDR11], the definition of Bayesian attack graph is first formulated. The authors propose Bayesian attack graphs as a method to encode the contribution of different security conditions during system compromise. Organisation's security risk is proposed to be calculated based on the metrics defined in the Common Vulnerability Scoring System (CVSS)<sup>2</sup>. According to the authors, the Bayesian attack graph consists of: (i) a set of attributes, an attribute being a Bernoulli random variable representing the state of some generic system property; (ii) a set of edges between attributes; (iii) a set of attribute and decomposition mappings, where decomposition can have a value AND or OR and defines whether all or any of parent attributes must be in `true` state for the attribute itself to be `true`; (iv) a set of discrete condition probability distribution functions representing probabilities of the attribute states based on the state of its parents.

In [GHL18], the authors propose a probabilistic model to perform the security risk assessment procedure. The framework takes a network asset and a vulnerability database as inputs and, through multiple phases, outputs the computed static security risk. The intermediate phases

---

<sup>2</sup><https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>



include: (i) *risk detection* - firstly, network assets are identified and, secondly, network hosts are scanned for vulnerabilities; (ii) *risk assessment* - network vulnerability, system configuration and host connectivity are inputs to the MulVAL tool [OGA<sup>+</sup>05]. The tool establishes a probabilistic attack graph according to the causal relationships among the multi-stage attacks using Bayesian networks. Leveraging inference algorithms, static risk assessment can be performed.

The increased threat that unknown vulnerabilities and the information asymmetry between attackers and defenders pose to the system is discussed in [HLL19]. The authors distinguish between Known Vulnerability Risk Assessment (KVRA) and Unknown Vulnerability Risk Assessment (UVRA). Similarly to the [GHL18] and to the methodology proposed in this work, the KVRA includes network information acquisition (based on vulnerability databases and scanners), vulnerability quantification (CVSS or other metrics), automatic attack graph generation and vulnerability risk assessment based on the generated graph. The UVRA divides vulnerabilities into known and zero-day by setting a time point. For example, if the time point is set as 2018/12/31, vulnerabilities before this point are considered known and vulnerabilities after this point are regarded as zero-day vulnerabilities.

### 3 Methodology

The risk assessment methodology used in this work is shown in Fig. 1. Four distinct steps must be performed:

1. **Ontology Development** To systematically gather knowledge an ontology must be defined which determines the required types of knowledge and relationships between known facts. In the overall risk assessment context, the developed ontology provides rules for the attack graph generation. These rules can be verified and their semantic reasoning discussed before any further risk assessment.
2. **Ontological Knowledge Acquisition** After the ontology is developed, an ontological knowledge about the target system must be gathered. Various automatic and manual procedures can be used for this goal: vulnerability scanners, consultations, model checkers, language models, static and dynamic code analysis etc. In the context of electronic voting, the knowledge is gathered from the information about the proposed system (source code, configuration, etc.), the vulnerability knowledge bases and legal regulations.
3. **Attack Graph Generation** Based on the gathered knowledge, and the developed ontology, an attack graph is generated. For efficiency and maximum correctness, this phase should be automatic. Automatically generated graphs can be manually pruned. However, the necessity of pruning is a symptom that the developed ontology lacks detail that would allow automatic filtering of such nodes. After this step the ontology could be refined.
4. **Risk Assessment** Many algorithms and automatic tools exist to efficiently infer Bayesian attack graphs and calculate unconditional probabilities of each node (state). Based on the calculation result the system analyst can identify the most probable attack paths, vulnerabilities that require attention and mitigation, the most vulnerable components.

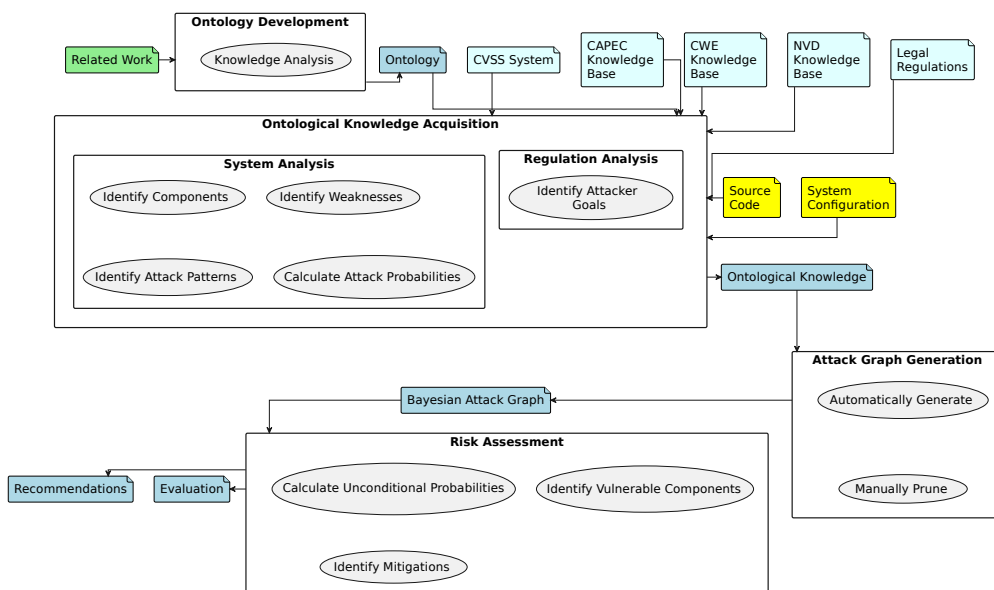


Fig 1. The risk assessment methodology.

### 3.1 Ontology Development

When dealing with knowledge, ontologies are irreplaceable. Ontology-driven threat modelling is a well developed concept. OWASP develop a framework called OdTM<sup>3</sup> that enables formalization of security related knowledge in form of domain-specific ontologies in the OWL (Web Ontology Language) format. The framework allows to describe a system with a data flow diagram, and use automatic reasoning procedures to build a threat model. An extension of the OWL language, called PR-OWL [CLC17], introduces new definitions to model uncertainties. According to [NM<sup>+</sup>01], ontology development is an iterative process which starts by defining the domain and scope. The scope can be defined by sketching a list of questions that the knowledge base based on the ontology should be able to answer. Additionally, concepts and their hierarchies must be defined. Ontologies are only true ontologies if concepts are related to other concepts (the concepts have attributes). Relations specify how objects are related to other objects. Much of the power of ontologies comes from the ability to describe relations. An important type of relation in the subsumption relation (*is a*) that defines classifications of objects. Another common type of relation is the mereology relation (*part of*) that represents how objects combine to form composite objects.

For this work, the ontology is developed so that an analyst can gather necessary information to assess risk. The current state of the ontology focuses on software vulnerabilities and does not allow for network protocol vulnerability representation nor for more complex types of communication.

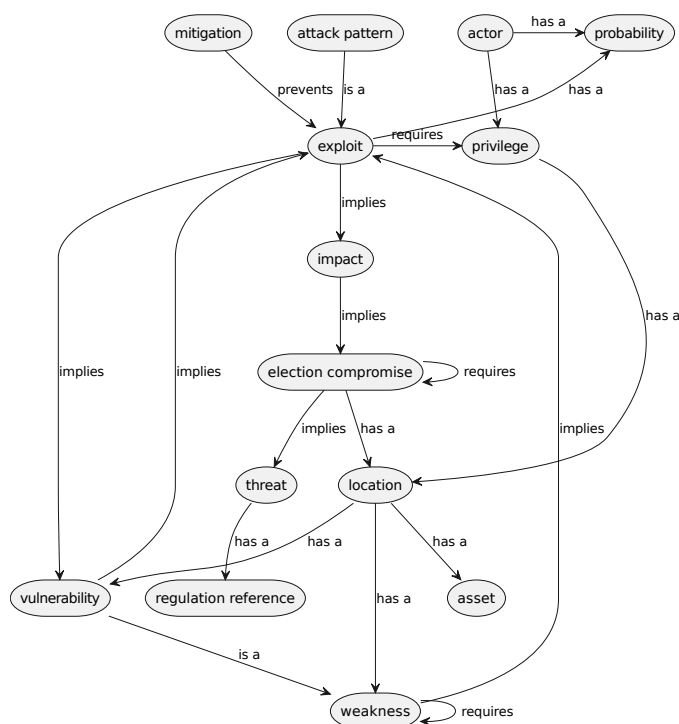


Fig 2. The developed ontology.

The developed ontology visualised in the Figure 2 defines five types of relationships:

- *has a* - a relationship between a state and its attribute: an state has an attribute.

<sup>3</sup><https://owasp.org/www-project-ontology-driven-threat-modeling-framework/>

- *is a* - a relationship between a state and its refinement: a state refinement is a state and inherits the state's attributes, preconditions and postconditions.
- *implies* - a relationship between a precondition and the postcondition: a precondition implies a postcondition.
- *prevents* - an opposite relationship to the *implies*: a precondition prevents a postcondition.
- *requires* - a relationship between a postcondition and its precondition: a postcondition requires a precondition.

The following are the knowledge types required for the analysis performed in this work:

- *asset* - a digital asset (a piece of data) that an attacker aims to obtain.
- *location* - a location (component) in the system's network.
- *actor* - any internal or external actor that uses or wishes to use (misuse) the system.
- *privilege* - rights in the system that an actor can have.
- *weakness* - a system's condition that, if exploited, may lead to vulnerabilities.
- *vulnerability* - an exploited weakness that have some undesired impact on the system's state.
- *impact* - a consequence that a successful exploitation of vulnerabilities has.
- *exploit* - an event of making use of weaknesses or vulnerabilities to reach some system state.
- *probability* - a likelihood of an event.
- *attack pattern* - a collection of concrete steps that an attacker can take to exploit a weakness or vulnerability.
- *election compromise* - a state where the election protocol run is compromised. Not every system compromise leads to the election compromise due to the possibility of security controls or fault tolerance.
- *threat* - a threat to system requirements.
- *mitigation* - a security control that minimises the probability of an exploit.
- *regulation reference* - a reference to the concrete legal regulation for requirement traceability.

## 3.2 Knowledge Bases

Most of knowledge-driven risk assessments rely on publicly maintained knowledge bases. This work retrieves data from the following enumerations:

- **Common Vulnerability Enumeration (CVE)**<sup>4</sup> - a list of publicly disclosed security flaws.
- **Common Weakness Enumeration (CWE)**<sup>5</sup> - a list of publicly maintained list of common software and hardware weaknesses. A weakness is a condition in a system which under certain circumstances could contribute to the introduction of vulnerabilities.
- **Common Attack Pattern Enumeration and Classification (CAPEC)**<sup>6</sup> - a publicly available catalogue of common attack patterns that helps understanding how adversaries exploit weaknesses and what impacts the exploits imply. CAPEC patterns are linked to the CWEs they exploit and, likewise, CWEs list CAPEC entries that they are susceptible to.

## 3.3 Common Vulnerability Scoring System

To quantify the analysis, some sort of numerical evaluation of the threats must be provided. *Common Vulnerability Scoring System (CVSS)*<sup>7</sup> is a free and open industry standard for assessing the severity of vulnerabilities. When in lack of more precise probability calculations, the analyses usually benefit from CVSS scores to approximate the risk. The CVSS score is a decimal number on a scale of 0 to 10 and consists of three metric groups: base, temporal and environmental. The base metrics quantify the intrinsic characteristics of a vulnerability with two subscores - (i) the exploitability subscore, composed of the access vector ( $B_{AV}$ ), access complexity ( $B_{AC}$ ), required privileges ( $B_{PR}$ ) and user interaction ( $B_{UI}$ ); (ii) the impact subscore, expressing the potential damage on confidentiality ( $B_C$ ), integrity ( $B_I$ ) and availability ( $B_A$ ). Following the example of the [PDR11], probability of a vulnerability  $v$  can be estimated from the exploitability attributes and the CVSS exploitability formula:

$$Pr(v) = 0.822 * B_{AV} * B_{AC} * B_{PR} * B_{UI} \quad (1)$$

## 3.4 Bayesian Attack Graphs

Attack graphs represent prior knowledge about vulnerabilities and network connectivity, enabling system administrators to reason about threats and their risks in a formal way. They are an efficient tool to *statically* analyse and determine the most effective threats and produce a better countermeasures selection. Two main types of attack graphs are used in the literature: state-based representations and logical attack graphs. In state-based representations, each node represents the

---

<sup>4</sup><https://nvd.nist.gov/>

<sup>5</sup><https://cwe.mitre.org/>

<sup>6</sup><https://capec.mitre.org/>

<sup>7</sup><https://www.first.org/cvss/user-guide>

state of the whole network after a simple atomic attack, and contains a table with global variables defining that state. In contrast, logical attack graphs are defined as bipartite graphs which represent dependencies between exploits and security conditions. In [MSB<sup>+</sup>17], logical representations are formally defined as a directed bipartite graph  $G = (E \cup C; R_r \cup R_i$ , where the vertices  $E$  and  $C$  are the sets of exploits and security conditions, respectively, and the edges  $R_r \subseteq C \times E$  and  $R_i \subseteq E \times C$  are *require* and *imply* relations.

In [PDR11], attack graphs were extended to support probabilistic modelling and a concept of Bayesian attack graphs was introduced. Bayesian attack graphs extend classical attack graphs by assigning every node with a local conditional probability distribution (LCPD). LCPD is a set of probability values specifying the chances of the node being compromised, given different combination of states of its parents. The existence of this probability is what primarily differentiates a Bayesian attack graph from a classical attack graph. Even if all preconditions of an attack have been met, there can still be a nonzero probability that an attacker is not able to carry out all the exploits successfully. Bayesian graph nodes have a decomposition property which determines how the conditional probabilities are calculated.

After the construction of a Bayesian attack graph, static risk assessment can be conducted by calculating the *unconditional probabilities* of each node. For these calculations, subjective probabilities of the initial state (which usually determines probabilities that attacker with specific capabilities exist) must be provided. There exist multiple algorithms and automatic tools to calculate unconditional probabilities due to the fact that the exact calculation following the Bayes rule is an NP-Hard problem that needs optimisation for bigger graphs.

Steps of the Bayesian attack graph static analysis:

1. Build the BAG: nodes represent the different security states (compromises) that an attacker can reach and different conditions (exploits) contributing to compromises with nonzero probabilities. For each node  $X_i$  there is a directed edge from each node in the set of parent nodes  $pa_i$  pointing to  $X_i$ . Each node  $X_i$  can have one of two states: 0, 1.
2. Determine the probability  $p_{v_i}$  of the successful exploitation of the  $v_i$  that leads to the state  $X_i$ , (e.g. using the base score metric of the CVSS score discussed in Section 3.3).
3. Build the conditional probability tables based on OR/AND node decomposition rules. As defined in literature, a logical AND (all preconditions should be met to compromise node  $X_i$ ) can be expressed as:

$$p(X_i|pa_i) = \begin{cases} 0 & \text{if } \exists X_j \in pa_i | X_j = 0 \\ \prod_{X_j=1} p_{v_j} & \text{otherwise} \end{cases} \quad (2)$$

A logical OR (only one of the preconditions in  $pa_i$  need to be satisfied to compromise  $X_i$ ) can be expressed as:

$$p(X_i|pa_i) = \begin{cases} 0 & \text{if } \forall X_j \in pa_i | X_j = 0 \\ 1 - \prod_{X_j=1} (1 - p_{v_j}) & \text{otherwise} \end{cases} \quad (3)$$

4. Compute the unconditional probabilities of all nodes in the BAG. These probabilities serve as risk estimates that can be used to detect weak areas in the network and serve as an input for network hardening or static risk mitigation techniques.

### 3.5 Automatic Graph Generation

An in-house tool was built around the *AgenaRisk* modeller library<sup>8</sup> to generate attack graphs. The *AgenaRisk* modeller is a design and execution environment for creating and analysing Bayesian networks. The developed tool derives network nodes, edges and node probability tables (NPT) based on the rules imposed by the ontology:

- knowledge items of all types except exploits (and attack patterns) define nodes in the network;
- exploits define edges between nodes, and their probabilities are used to calculate the NPTs of nodes based on the rules described in the Section 3.4;
- attack graph starts from the actor type nodes, and following the *implies* and *prevents* relationships reaches threat type nodes which are terminal;
- mitigation nodes are special in the sense that they set the exploit's probability to zero for all postcondition's that they prevent;
- the graph is pruned so that only the complete paths between actor and threat nodes remain.

### 3.6 Risk Assessment

After the network is constructed, the *AgenaRisk* environment allows for several types of analyses. Observational information can be entered in place of the probabilities in the NPT. By doing this, what-if scenario analysis can be done to examine the effects specific risk events have on threat probabilities. Sensitivity studies can be conducted to determine a critical path of influence in a risk network.

---

<sup>8</sup><https://pypi.org/project/pyagena/>

## 4 Ontological Knowledge Acquisition

### 4.1 Regulation Analysis

In this work, the feasibility of an Internet voting system is evaluated in the context of legal Internet voting regulations in Lithuania. The regulations are informally mapped onto the ontological concept of a threat. As per the developed ontology, threats are linked to the source regulation for traceability. Threat knowledge table is provided in the 2.

Legal Internet voting regulations, discussed in Section 2.1.2, are defined in the Internet voting draft law proposed in 2018 by the Ministry of Justice of the Republic of Lithuania [Min18]. Based on the regulations and the related security analyses ([SFD<sup>+</sup>14], [JRS<sup>+</sup>04]), five major threats are formulated:

1. **Voters' Disenfranchisement** Internet voting provides opportunities for selective disenfranchisement, either of individuals or of classes of voters based on the likelihood of their preference. The threat could be fulfilled by a denial of service attack, or a malicious component in the voting client detaining the votes. These kinds of attacks require low skill and can be hard to diagnose. Denial of service attacks can be prevented by elimination of any single points of failure that could be attacked. Preventing voters from casting a ballot, or ballot box from recording the ballot would directly disobey the Article §3.3 of the draft law.
2. **Ballot Altering** Electronic voting faces the threat of an attacker not only detaining but also modifying the ballot. As discovered in the Estonian e-voting system security analysis [SFD<sup>+</sup>14], a malicious component in the voter's device can change the vote after the vote verification phase has ended and the voter has no way of verifying their submission. This kind of attack could be deployed widely and completely compromise the election. Ballot forgery would make the system non compliant with the Articles §9.1, §9.2 and §9.3 of the draft law.
3. **Voter Privacy Breach** The secrecy of voters' ballots is a crucial asset in an electronic voting which can attract attackers of various skills. In a traditional paper-based e-voting protocol, voters' privacy is protected by the voting booth which isolates the voter, and a mixed ballot box which separates the ballot from its owner in an untraceable manner. Most electronic voting frameworks protect voter privacy with encryption. However, even without the cryptographic material, side-channel attacks are a powerful threat to private information. Protection of the voter's privacy is defined the Article §3.1 of the draft law.
4. **Multiple Ballots** A well-known voter fraud is when a voter votes more than once. The feasibility of this threat depends on the voter identification and authorisation correctness and the duplicate ballot removal implementation in the tally stage. Firstly, an e-voting protocol must have a way to anonymously identify ballots belonging to the same voter. Only after the duplicate ballot removal can the system strip this information and shuffle



the depersonalised ballots. Adding additional ballots to the ballot box would breach the Article §3.3 in the draft law.

5. **Non Eligible Voters** Articles §6.1 and §9.4 regulate that only eligible voters must be able to participate in the election process. In the paper and electronic voting, eligible voters are listed out in the electoral roll. Secure storage and eligibility verification procedures are crucial to the integrity of the ballot box.

<b>Id</b>	<b>Name</b>	<b>Regulation Reference</b>
threat-1	Ballot Altering	§9.1 §9.2 §9.3
threat-2	Voter Privacy Breach	§3.1
threat-3	Voter Disenfranchisement	§3.3
threat-4	Multiple Ballots	§3.3
threat-5	Non Eligibile Voters	§6.1 §9.4

Table 2. Knowledge of threats.

## 4.2 System Analysis: *Chirotonia*

*Chirotonia* is an e-voting platform developed and maintained by the Network Security research group at the University of Naples Federico II [RAV<sup>+</sup>21]. It offers a framework for small-scale auditable e-voting with low-risk coercion. In addition to being a theoretic sandbox for the research group, *Chirotonia* is successfully used in local university elections. The security of the platform is claimed to be guaranteed by the blockchain as a secure, untamperable, auditable, distributed ledger, and linkable ring signatures as an anonymous proof of voters' eligibility and vote uniqueness. According to the *Chirotonia* paper [RAV<sup>+</sup>21], the chain-of-blocks data structure allows any modification to the system state to be audited in an untamperable manner on the blockchain itself as part of the state (e.g. a transaction that calls a function in a smart contract is stored including information on the called function and all the parameters passed as arguments). These features are claimed to make the platform highly auditable and transparent to anyone. Another important building block of *Chirotonia* are linkable ring signatures [LWW04]. They allow the creation of signatures on behalf of a group of signers while hiding the identity of the actual signer among the group. The ring signature can be publicly verified to have been produced by a member of the provided group. Additionally, the linkability feature introduces a *public* way of determining whether two signatures have been produced by the same signer while preserving the anonymity of the signer and the signed message (ballot).

### 4.2.1 Assets

During the run of the *Chirotonia* voting protocol, multiple assets are created and transmitted between system components. The Table 3 lists out the identified assets and assigns a symbol for traceability in diagrams.

Assets can be viewed as groups: (i) election protocol assets; (ii) cryptographic material required by chosen cryptographic schemes; (iii) authorisation assets.












<b>Id</b>	<b>Name</b>	<b>Symbol</b>
asset-1	Plain Text Ballot	
asset-2	Encrypted Ballot	
asset-3	Signing Key	
asset-4	Signature Verification Key	
asset-5	Decryption Key	
asset-6	Session Cookie	
asset-7	Electoral Roll	
asset-8	Voting Card	
asset-9	User's Password	
asset-10	Encrypted Signing Key	
asset-11	Ethereum Account Private Key	N/A
asset-12	Voter's Signature	

Table 3. Knowledge of assets.

**Election Protocol Assets** Assets required by the *Chirotonia* voting protocol are similar to the ones found in similar systems and the paper-based voting protocols. All voting is done via ballots, the voter eligibility is traced in the electoral roll, and the registration and voting phases are separated by the introduction of voting cards that are the outcome of successful registration.

**Cryptographic Material** *Chirotonia* encrypts ballots with the RSA-2048 scheme which, in the current implementation, yields a 2048-bit decryption key asset. Theoretically, this key is supposed to be generated by distributed actors, but this is not the case in the analysed version of the system.

After encryption, ballots are signed with a linkable ring signature. The signature scheme is defined by four probabilistic polynomial-time algorithms ( $KeyGen, Sign, Verify, Link$ ) such that:

- $KeyGen(1^l)$ , the key generation algorithm, takes as an input the security parameter  $l$  and outputs a pair  $(pk, sk)$  of public (verification) and secret (signing) keys;
- $Sign(1^l, sk_\pi, pk_1, \dots, pk_n, m)$ , the signing algorithm, takes as input the security parameter  $l$ , a secret key  $sk_\pi$  for some  $\pi \in \{1, \dots, n\}$ , a message  $m$  and a list of public keys  $pk_1, \dots, pk_n$ , and outputs a signature  $\delta$ ;
- $Verify(1^l, pk_1, \dots, pk_n, m, \delta)$ , the verification algorithm, takes as input the security parameter  $l$ , a list of public keys  $pk_1, \dots, pk_n$ , a message  $m$ , and a signature  $\delta$ , and outputs a bit  $b \in \{0, 1\}$ : 1 if the signature is recognised as valid w.r.t. the list of public keys, and 0 otherwise;
- $Link(\delta_1, \delta_2, m_1, m_2)$ , the linking algorithm, takes as an input two signatures,  $\delta_1, \delta_2$  and two messages  $m_1, m_2$ , and outputs a bit  $b \in \{0, 1\}$ : 1 if the two signatures have been produced with the same secret key, 0 otherwise.

The signature scheme yields the signing key and the signature verification keys. *Chirotonia* was designed to store the signing key in the voting card which requires encryption. The signing key is encrypted with the AES-256 scheme for which the key is derived from the user-inputted password.

**Authorisation Assets** The main authorisation asset is the session cookie that the client receives after successful authorisation. Additionally, the underlying administrative Ethereum account's have their private keys.

#### 4.2.2 Locations

*Chirotonia* is composed of multiple voting and administration components that play their role in the run of the protocol. The Table 4 lists out the components that are cross-referenced with the assets they have access to.

<b>Id</b>	<b>Name</b>	<b>Assets</b>
loc-1	Voting Server	asset-2, asset-8, asset-4, asset-6
loc-2	Voting Client	asset-1, asset-2, asset-4, asset-3, asset-6, asset-8, asset-9, asset-10
loc-3	Ethereum Signer	asset-11, asset-2, asset-4
loc-4	Database	asset-2, asset-7, asset-4
loc-5	Ethereum Validator	asset-2, asset-4
loc-6	Ethereum Node	asset-2, asset-4
loc-7	Public Network	asset-2, asset-4
loc-8	Smart Contract	asset-4, asset-2
loc-9	Administration Server	asset-5
loc-10	Voting Device	asset-8
loc-11	Ethereum Explorer	asset-2, asset-4

Table 4. Knowledge of locations.

1. **Voting Clients** Voters' browsers running the ReactJS voting applications. During different phases have access to the session cookie, plain text ballot, encrypted ballot, plain text and encrypted signing key, signature verification key, voting card and user's password.
2. **Voting Devices** Devices on which the voters vote. The devices store voters' voting cards.
3. **Chirotonia Web Server** A Node.js restful web server responsible for registering voters, checking their eligibility, validating and writing the ballots to the blockchain. During different phases have access to the encrypted ballots, signature verification keys, voting cards and session cookies.
4. **Blockchain Network** A private and permissioned Ethereum network. Consists of:
  - **Ethereum Node** A Hyperledger Besu Ethereum client to which all the transactions are sent by the web server. During different phases has access to signature verification keys, encrypted ballots and, after the tally, the plain text ballot.
  - **Ethereum Validator** A Hyperledger Besu Ethereum client. It has access to the data provided in the transactions: signature verification keys, encrypted ballots and, after the tally, the plain text ballot.

- **Ethereum Explorer** A web application connected to the Ethereum Node. It provides access to Ethereum transactions and the data in them (signature verification keys, encrypted ballots, and the plain text ballot). Ethereum Explorer is meant to be used for auditing and verification purposes, however the blockchain data is hardly human-readable.
  - **Ethereum Signer** A proxy service responsible for signing blockchain transactions and forwarding them to the Ethereum client. Stores a private Ethereum account key. Being a proxy service, it has access to assets transmitted in the transactions: signature verification keys and encrypted ballots.
5. **Chirotonia Smart Contract** Self-executing *Chirotonia* protocol on the blockchain. Stores signature verification keys and the encrypted ballot box.
  6. **Database** NoSQL database which stores the electoral roll, signature verification keys, and the encrypted ballot box.
  7. **Identity Provider** A trusted identity provider used to authenticate voters. The most important asset associated with authentication is the session cookie.
  8. **Public Network** A communication channel through which the data from one component reaches another.

Figure 3 demonstrates communication between *Chirotonia* components.

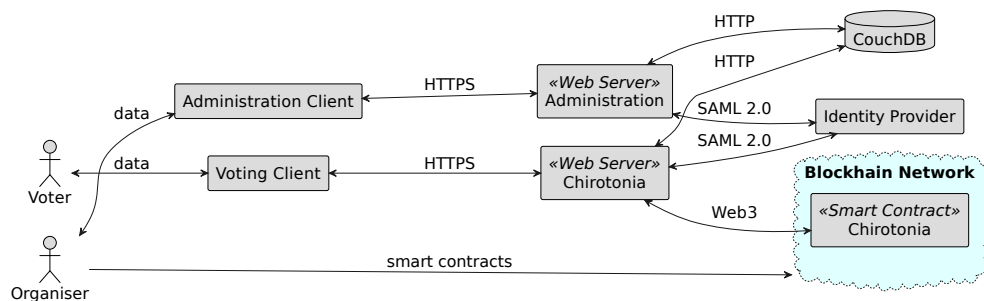


Fig 3. Communication between *Chirotonia* components.

#### 4.2.3 Phases

From the voter's perspective, *Chirotonia* runs through the registration, voting and tally phases. The *Chirotonia* paper [RAV<sup>+</sup>21] envisions the phase transitions enforced by the blockchain events. In the implementation, vote opening/closing blockchain events are manually fired by the organiser. The blockchain will not accept voter identifiers nor ballots during improper states. Additionally, the voting server has its own election state tracking in the database which is updated manually by the organiser before each new phase.

From the organiser's perspective, additional phases of election setup, vote opening and closing need to be performed.

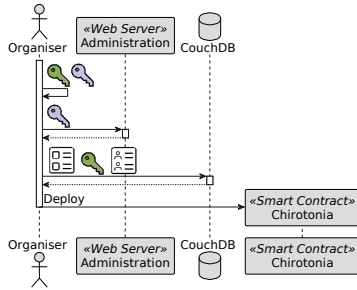


Fig 4. The setup phase sequence.

**Setup** The organiser initiates the election session by deploying the components and the smart contract. A pair of cryptographic keys is generated for each election. Before moving to the registration phase, the organiser manually saves the electoral roll and the election session data in the database. The sequence diagram in Figure 4 shows the flow of the setup phase.

**Registration** Before the voting, voters are asked to authenticate themselves via the identity provider through the use of the SAML 2.0 protocol. The protocol sequence can be seen in Figure 5. Before accessing any resources, a user is redirected to the identity provider to authenticate. After successful login, the identity provider generates and sends an authentication token to the user's browser. Any subsequent request by the user to the web server will contain the generated token which can be used to verify the user's identity.

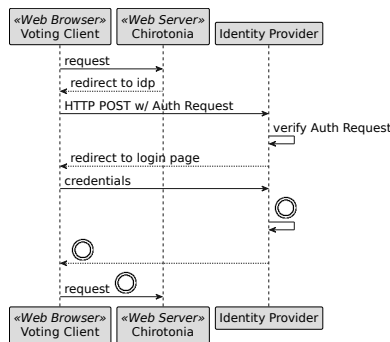


Fig 5. SAML 2.0.

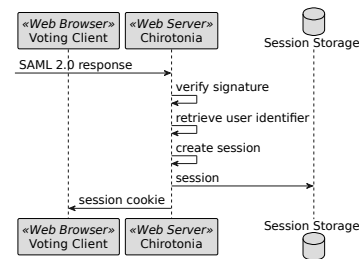


Fig 6. Session storage.

In the *Chirotonia* server implementation, the authentication tokens are handled by the `passport-saml` library. After receiving a SAML 2.0 response, its signature is verified against the configured Identity Provider's public certificate. If the SAML response is valid, the web server stores the user's identifier in a session. User sessions are managed by the `express-session` library which creates and sends a session cookie to the client's browser. For each subsequent request, the browser sends the session cookie back to the server which validates it against a session store (in-memory, or database). A valid session cookie is the only asset needed to access server resources. Session configuration in *Chirotonia* does not set the maximum cookie age.

After the successful authentication, the web server additionally verifies the voter's eligibility against the electoral roll. Upon approval from the web server, the voters can register themselves for an election by generating a voting card which includes the voter's signing and verification keys and the signed identifier (e-mail address). The signing key is encrypted with a key derived from a user-inputted password, and the voting card is sent to the web server. The voter must download their card to their device, and, optionally, save it in the browser's local storage. [Pis21]. The sequence diagram in Figure 7 shows the flow of the registration phase.

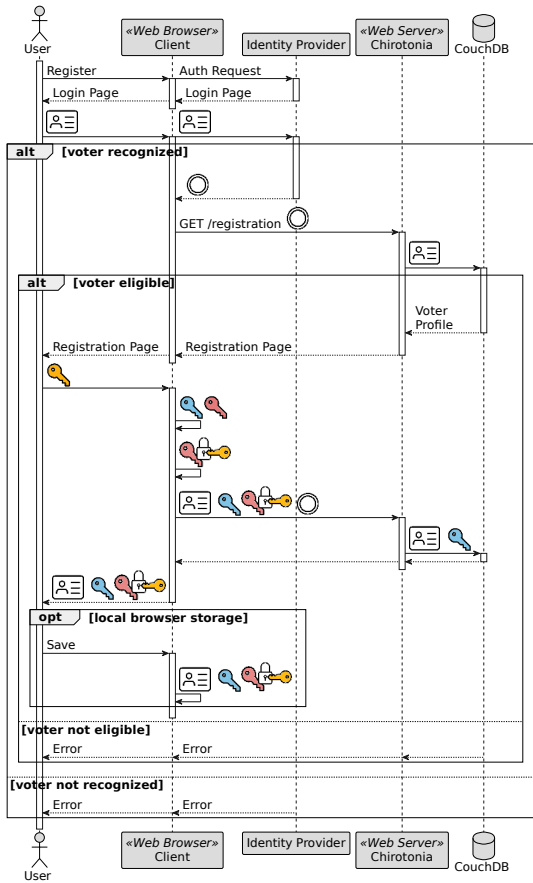


Fig 7. The registration phase sequence.

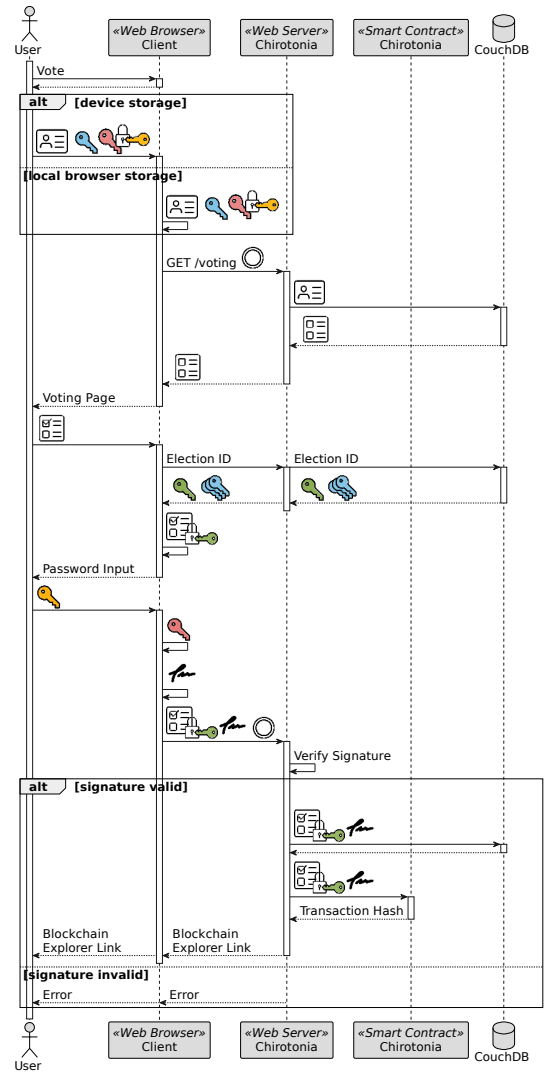


Fig 9. The voting phase sequence.

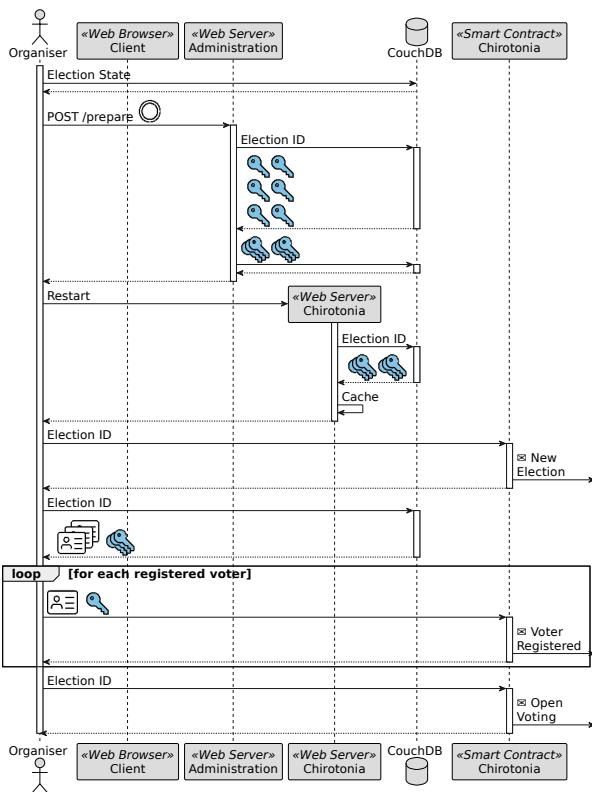


Fig 8. The opening phase sequence.

**Opening** After the registration phase is over, the organiser divides registered voters into signing groups and stores the election data in the *Chirotonia* smart contract. To open the voting phase, the organiser manually updates the election state in the database and fires the Open Voting blockchain event. Linkable ring signature schemes are created and cached in the voting server for optimised signature verification. The sequence diagram in Figure 8 shows the flow of the opening phase.

**Voting** During the voting phase, a voter uploads the voting card to the browser (if it hasn't been saved in the local browser storage during the registration phase), selects their preference, and inserts the password to decrypt the signing key. The browser calculates the linkable ring signature, encrypts the ballot, and sends them both to the voting server which validates the signature, saves the ballot to the database and transacts the ballot to the blockchain. The voting server returns to the voter the blockchain explorer link as a receipt. The sequence diagram in Figure 9 shows the flow of the voting phase.

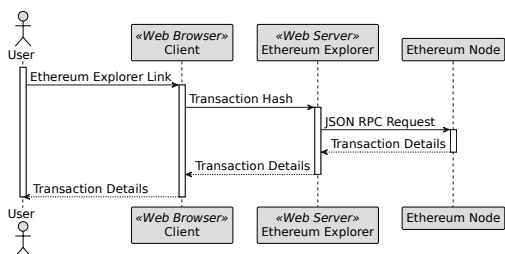


Fig 10. The vote verification phase sequence.

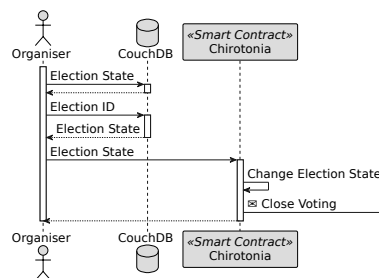


Fig 11. The closing phase sequence.

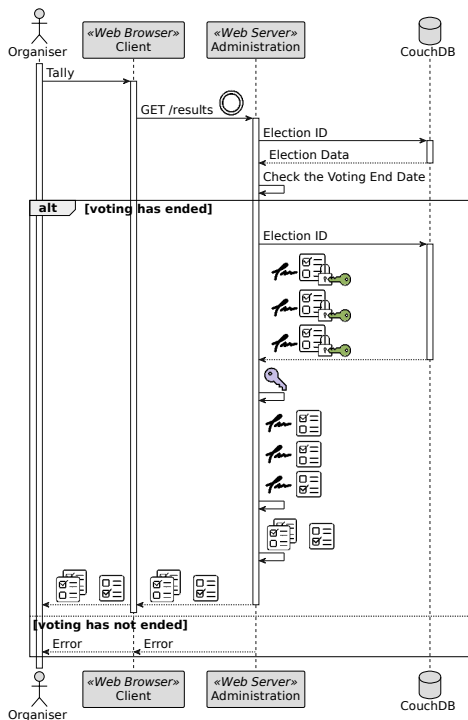


Fig 12. The tally phase sequence.

**Verification** Voters can verify their ballot transaction details with the blockchain explorer link provided to them after successful voting. The transaction details contain the transaction date, sender and the encrypted ballot. The ballot decryption key is only released after the voting phase has been closed so there is no way to verify the decrypted ballot during the voting. The sequence diagram in Figure 10 shows the flow of the verification phase.

**Closing** To end the election session, the organiser closes the voting phase in the database and fires the Close Voting blockchain event. The sequence diagram in Figure 11 shows the flow of the closing phase.

**Tally** The election tally is calculated in the administration server by loading all valid ballots from the database, decrypting them with the decryption key

(stored in the server during the setup phase), and grouping valid ballots by the chosen candidate. Election outcome and the log of all ballot transaction hashes can be exported to a file. Signatures are not re-validated during the tally calculation. Ballots stored in the smart contract are ignored during the tally performed by the administration portal. To verify that the published outcome is correct, one would need to audit the ballot transaction hash log manually or automatically. The sequence diagram in Figure 12 shows the flow of the tally phase.

#### 4.2.4 Election Compromises

Based on the system analysis possible election compromises are identified. As per the ontology, compromises happen in locations that have target assets. Impacts on specific assets imply compromises, and compromises imply threats to regulation compliance. Compromises can be composite and have preconditions. Table 5 lists out prominent election compromises.

<b>Id</b>	<b>Name</b>	<b>Asset</b>	<b>Impact</b>	<b>Threat</b>	<b>Requires</b>
comp-1	Falsely Submitted Ballot			threat-1	comp-2, comp-3, comp-9
comp-2	Stolen Password	asset-9	Confidentiality:Read Data		
comp-3	Stolen Voting Card	asset-8	Confidentiality:Read Data		
comp-4	Ballot Forgery	asset-2	Integrity:Modify Data	threat-1	comp-9
comp-5	Stolen Decryption Key	asset-5	Confidentiality:Read Data	threat-2	
comp-6	Plain Text Ballot Exposure	asset-2	Confidentiality:Bypass Protection Mechanism	threat-2	
comp-7	Unavailable Component	asset-2	Availability:Resource Consumption	threat-3	
comp-8	Ballot Box Write Access	asset-2	Access_Control:Gain Privileges	threat-3	
comp-9	Fake Receipt	asset-12	Integrity:Modify Data		
comp-10	Voting for an Absentee			threat-4	comp-11
comp-11	Stolen Session	asset-6	Confidentiality:Read Data	threat-1	
comp-12	Add Ineligible Authenticated Voters	asset-7	Integrity:Modify Data	threat-5	
comp-13	Add Fake Voters			threat-5	comp-14, comp-15
comp-14	Modify Electoral Roll	asset-7	Integrity:Modify Data		
comp-15	Bypass Authentication	asset-6	Authorization:Gain Privileges		
comp-16	Plain Text Ballot Exposure	asset-2	Confidentiality:Gain Privileges	threat-2	
comp-17	Dropped Ballots	asset-2	Integrity:Modify Data	threat-3	

Table 5. Knowledge of compromises.

#### 4.2.5 Weaknesses

This work assesses the risk of software weaknesses in the Voting Server and the Voting Client. The *Chirotonia* source code was scanned with vulnerability scanners to compile the vulnerabilities' knowledge table 6. The identifiers and additional information was retrieved from the knowledge bases as described in Section 3.2. Additionally, some weaknesses were added manually: the CWE-494 (Download of Code Without Integrity Check) weakness to assess the risk posed by a weak supply chain, the CWE-284 (Improper Access Control) to model a scenario where an attacker has access to the user's device, and the CWE-300 (Channel Accessible by Non-Endpoint) to assess network-related threats.

#### 4.2.6 Mitigations

Due to the lack of blockchain-specific weakness enumeration, vulnerabilities for blockchain components are not included in this analysis. Blockchain components are assumed to be secure and blockchain procedures in the protocol are considered as mitigations. However, in the current implementation of *Chirotonia*, the web server has many responsibilities and very little verification is



<b>Id</b>	<b>Name</b>	<b>Location</b>	<b>Requires</b>	<b>Patterns</b>	<b>Probability</b>	<b>Privileges</b>
CWE-441	Unintended Proxy or Intermediary ('Confused Deputy')	loc-1		CAPEC-141, CAPEC-142, CAPEC-219, CAPEC-465	0.39	NONE
CWE-770	Allocation of Resources Without Limits or Throttling	loc-1		CAPEC-125, CAPEC-130, CAPEC-147, CAPEC-197, CAPEC-229, CAPEC-230, CAPEC-231, CAPEC-469, CAPEC-482, CAPEC-486, CAPEC-487, CAPEC-488, CAPEC-489, CAPEC-490, CAPEC-491, CAPEC-493, CAPEC-494, CAPEC-495, CAPEC-496, CAPEC-528	0.31	NONE
CWE-613	Insufficient Session Expiration	loc-1			0.29	NONE
CVE-2022-39300	CVE-2022-39300	loc-1		CAPEC-463	0.22	NONE
CWE-352	Cross-Site Request Forgery (CSRF)	loc-1	CWE-346, CWE-441, CWE-642, CWE-613	CAPEC-111, CAPEC-462, CAPEC-467, CAPEC-62	0.28	NONE
CVE-2022-25896	CVE-2022-25896	loc-1		CAPEC-31, CAPEC-39, CAPEC-60, CAPEC-61, CAPEC-59, CAPEC-21, CAPEC-196	0.22	NONE
CWE-642	External Control of Critical State Data	loc-1		CAPEC-21, CAPEC-31		
CWE-384	Session Fixation	loc-1	CWE-346, CWE-472, CWE-441	CAPEC-196, CAPEC-21, CAPEC-31, CAPEC-39, CAPEC-59, CAPEC-60, CAPEC-61	0.29	NONE
CWE-472	External Control of Assumed-Immutable Web Parameter	loc-1		CAPEC-146, CAPEC-226, CAPEC-31, CAPEC-39	0.39	NONE
CWE-346	Origin Validation Error	loc-1		CAPEC-111, CAPEC-141, CAPEC-142, CAPEC-160, CAPEC-21, CAPEC-384, CAPEC-385, CAPEC-386, CAPEC-387, CAPEC-388, CAPEC-510, CAPEC-59, CAPEC-60, CAPEC-75, CAPEC-76, CAPEC-89	0.29	NONE
CVE-2023-44487	CVE-2023-44487	loc-1		CAPEC-197, CAPEC-147, CAPEC-492	0.39	NONE
CVE-2022-29078	CVE-2022-29078	loc-2		CAPEC-77, CAPEC-35, CAPEC-242	0.39	NONE
CWE-295	Improper Certificate Validation	loc-2		CAPEC-459	0.28	NONE
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	loc-2		CAPEC-209, CAPEC-588, CAPEC-591, CAPEC-592, CAPEC-63, CAPEC-85	0.09	HIGH
CWE-521	Weak Password Requirements	loc-2		CAPEC-112, CAPEC-16, CAPEC-49, CAPEC-55, CAPEC-70	0.34	NONE
CWE-300	Channel Accessible by Non-Endpoint	loc-7		CAPEC-466, CAPEC-57, CAPEC-589, CAPEC-590, CAPEC-612, CAPEC-613, CAPEC-615, CAPEC-94	0.21	NONE
CWE-494	Download of Code Without Integrity Check	loc-10		CAPEC-184, CAPEC-185, CAPEC-186, CAPEC-187, CAPEC-533	0.27	NONE
CWE-284	Improper Access Control	loc-10		CAPEC-19, CAPEC-441, CAPEC-478, CAPEC-479, CAPEC-502, CAPEC-503, CAPEC-536, CAPEC-546, CAPEC-550, CAPEC-551, CAPEC-552, CAPEC-556, CAPEC-558, CAPEC-562, CAPEC-563, CAPEC-564, CAPEC-578	0.3	LOW

Table 6. Knowledge of vulnerabilities.

actually performed in the blockchain. The smart contract is invoked twice: after the registration phase voter identifiers and signature verification keys are sent to the smart contract; and during the voting phase ballots are verified in the web server and sent to the smart contract. None of these invocations mitigate any damage done by vulnerable web server or the client. The biggest shortcoming is that the tally is performed on the ballot box stored in the database, not the smart contract. However, this could be easily mitigated in the future.

## 5 Risk Assessment

Risk assessment is performed by generating Bayesian attack graphs for each identified threat and calculating the probabilities of threats given specific risk scenarios with the *AgenaRisk* tool. *AgenaRisk* uses the Junction Tree algorithm to extract marginalization. Actors considered for the analysis are provided in the Table 7. The probabilities are assigned based on subjective assumptions.

<b>Id</b>	<b>Name</b>	<b>Privileges</b>	<b>Locations</b>	<b>Probability</b>
act-1	Remote Attacker	NONE		0.7
act-2	Malicious Insider	HIGH	loc-1, loc-3, loc-5, loc-6	0.25

Table 7. Knowledge of actors.

### 5.1 Ballot Altering

In comparison with graphs for other threats, the ballot altering threat has the most possible pre-conditions in the current system implementation, and many different attack paths could be taken. As per the system analysis, two distinct election compromise scenarios lead to altered ballots: (i) falsely submitting ballots - includes stealing the voter's voting card and the password; (ii) forging the ballot (exploiting integrity vulnerabilities in the system). The Figure 13 shows the ballot altering probability under a risk scenario when a remote attacker is trying to alter ballots and no vulnerabilities have been mitigated. We can observe that it is unlikely that the ballot will be altered exploiting vulnerabilities in the public network: stealing the voter's session or forging the ballot in the network is hard.

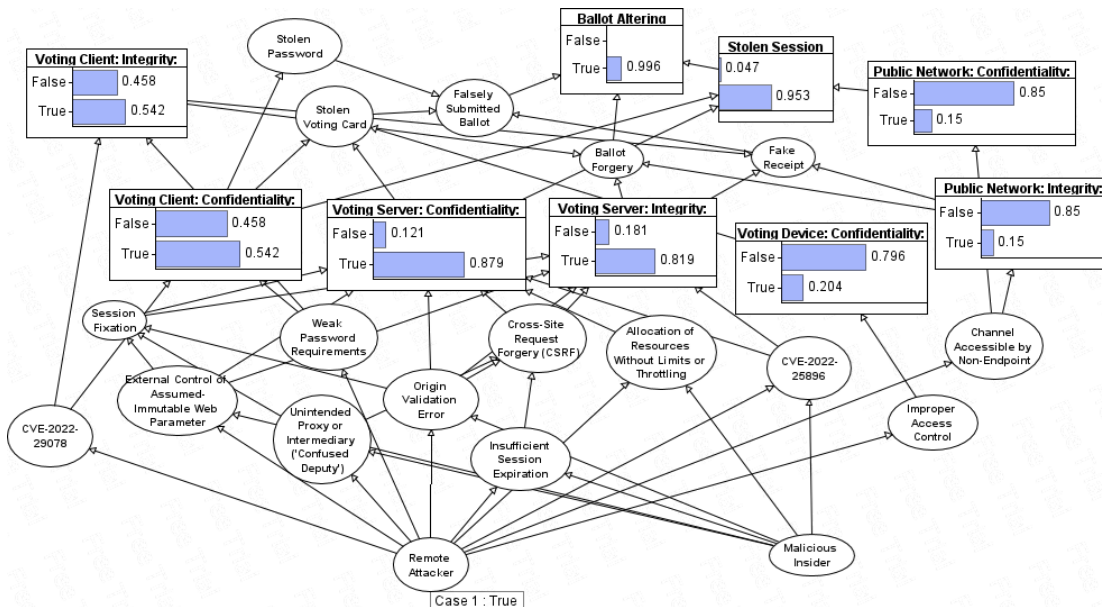


Fig 13. Calculation of the risk scenario when a remote attacker is trying to alter ballots.

To evaluate the likelihood of the stolen session due to vulnerabilities in the voting server, we can analyse a different risk scenario shown in the 14. Here the confidentiality of the voting client and the public network are set to be secured. The probability of a stolen session is still

high. This is due to prominent vulnerabilities in the web server, mainly the session fixation and the composite weaknesses that allow cross-site request forging.

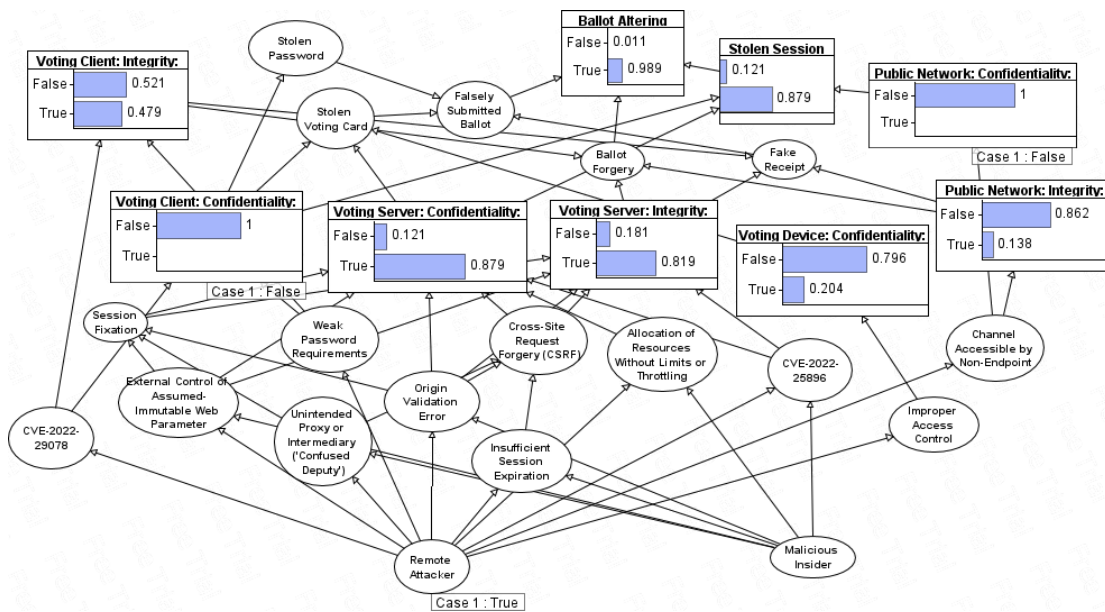


Fig 14. Calculation of the risk scenario when a remote attacker is trying to alter ballots but the voting client's and public network's confidentiality have been secured.

The Figure 15 demonstrates that vulnerabilities in the voting server contribute the most to the overall likelihood of ballot altering. Prominent vulnerabilities, such as session fixation, allocation of resources without limits, cross-site request forging must be mitigated.

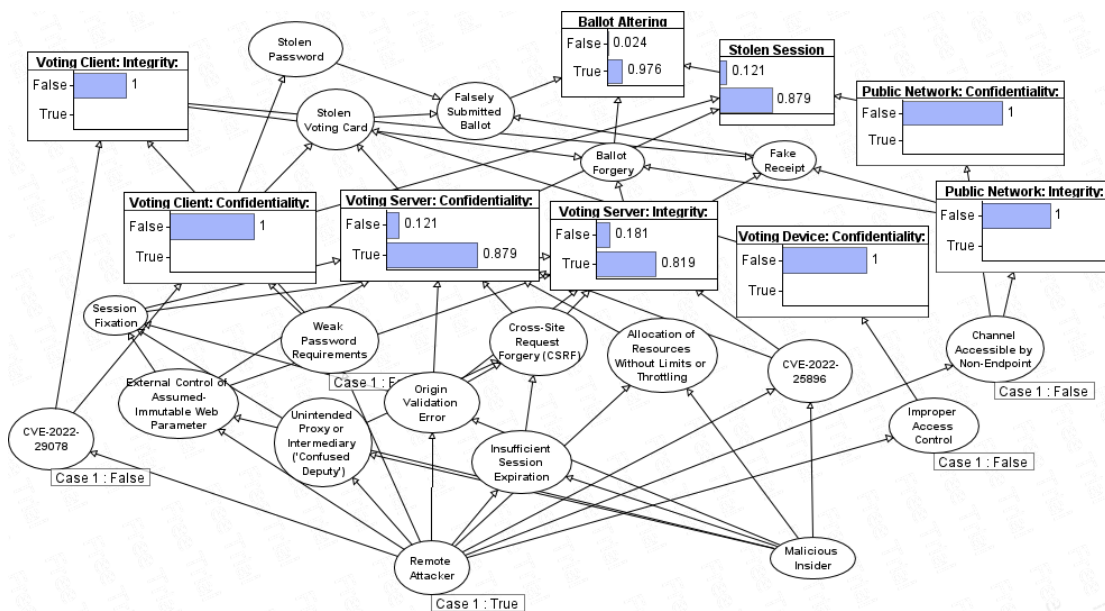


Fig 15. Calculation of the risk scenario when a remote attacker is trying to alter ballots and all relevant network and voting client vulnerabilities have been mitigated.

## 5.2 Voter Privacy Breach

Due to the fact the the decryption key is stored in the administration server which is not analysed in this work, the voter privacy breach threat could only be executed by exposing the plain text

ballot to unauthorised parties. One of the possibilities are the side-channel attacks, however the current state of attack graph modelling does not represent hardware vulnerabilities. Another, software-based compromise, is possible attacking the confidentiality of the voting client (and the device itself). The CVE-2022-29078 vulnerability, detected by vulnerability scanners in the supply chain of the voting client application, allows executing arbitrary OS commands if successfully exploited. The vulnerability could be exploited to track the voter's actions. However, as observed in the Figure 16 the more likely attack path is gaining the root access to the voting device itself and employing a number of techniques (e.g. keylogging) to discover the voter's ballot choices.

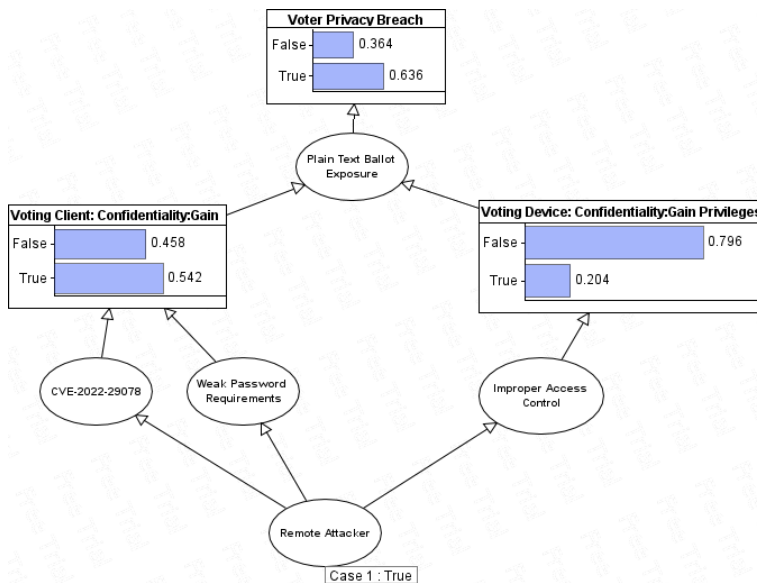


Fig 16. Calculation of the risk scenario when a remote attacker is trying to breach the voter's privacy.

### 5.3 Voter Disenfranchisement

Voter disenfranchisement, or ballot removal, is a likely threat. Three general attack paths can be observed in the Figure 17: (i) a denial-of-service attack on the voting server, not very likely based on the software vulnerabilities only; (ii) a man-in-the-middle attack to drop ballots in transit, most likely to be executed in the voting server due to the amount of vulnerabilities; (iii) unauthorised access to the ballot to remove the already persisted ballot, shown to be very likely in the current implementation. The third attack path could be completely mitigated by the proper usage of the blockchain which is considered to be untamperable. However in the current implementation, the tally is performed on the ballot box in the database with no verification against the smart contract storage, thus allowing successful voter disenfranchisement.

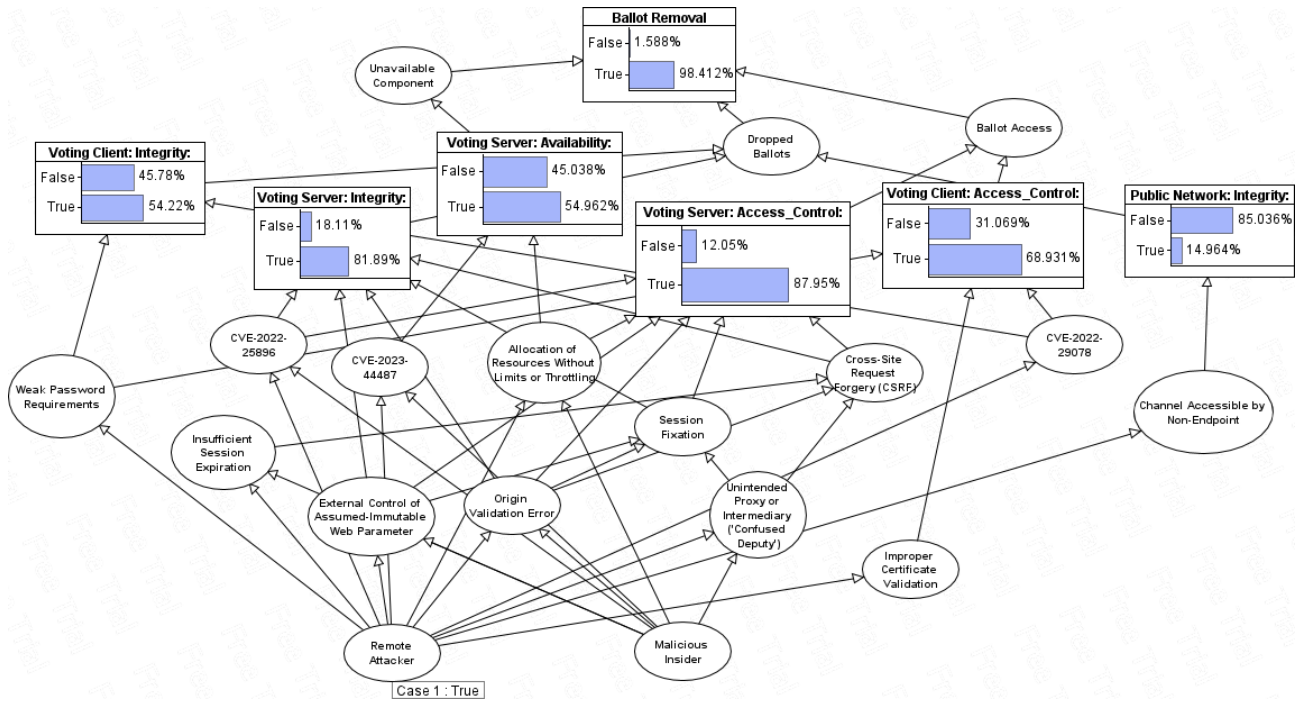


Fig 17. Calculation of the risk scenario when a remote attacker is trying to remove ballots.

## 5.4 Multiple Ballots

Based on the system analysis, adding illegitimate ballots in *Chirotonia* are possible only by voting for eligible but absent voters. According to the Figure 18, a remote attacker could potentially steal absent voters' sessions given that the voter has authenticated via the same identity provider in the same browser (which is not unlikely in single sign-on authentication mechanisms). The graph shows that the attack on vulnerable network's confidentiality is the most probable path to steal the voter's session. This could be achieved by the man-in-the-middle attack intercepting the communication and extracting session information from the requests. For this to be successful, a legitimate-looking certificate should be provided by the adversary proxy to the voting client. Due to the variety of computer literacy among the possible voters, the forged certificate could very likely go unnoticed.

## 5.5 Non Eligible Voters

Two types of possible non-eligible voters exist: (i) fake voters, which requires adding fake identifiers to the electoral roll and breaking the authentication mechanism; (ii) legitimate authenticated identities that are not eligible for the current election. Adding legitimate authenticated identities to the electoral roll is an easier attack path which can be executed in the voting server by, for example, a malicious code bypassing the electoral roll verification. This is represented in the Figure 19 as well, showing that the attack on the voting server's integrity is more likely than bypassing the authorization in the voting client.

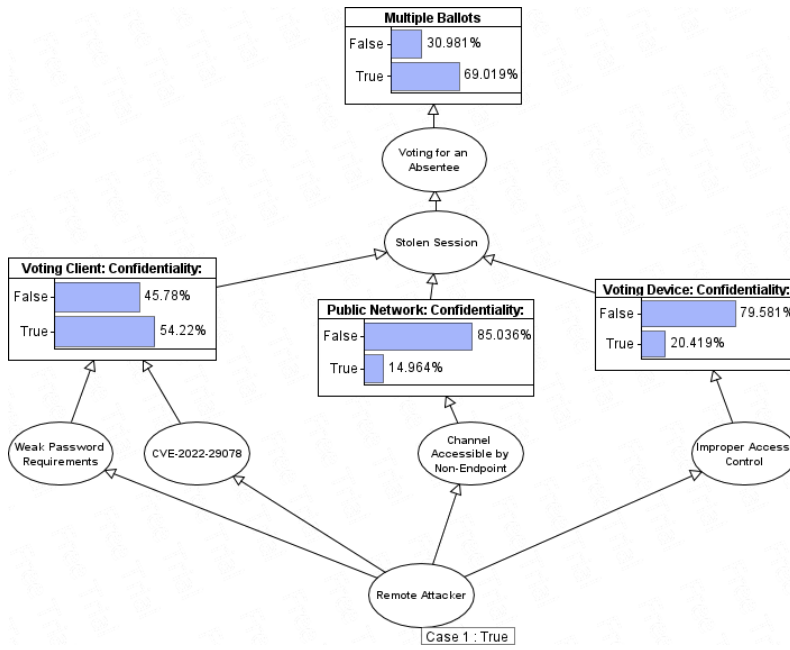


Fig 18. Calculation of the risk scenario when a remote attacker is trying to vote for absent voters.

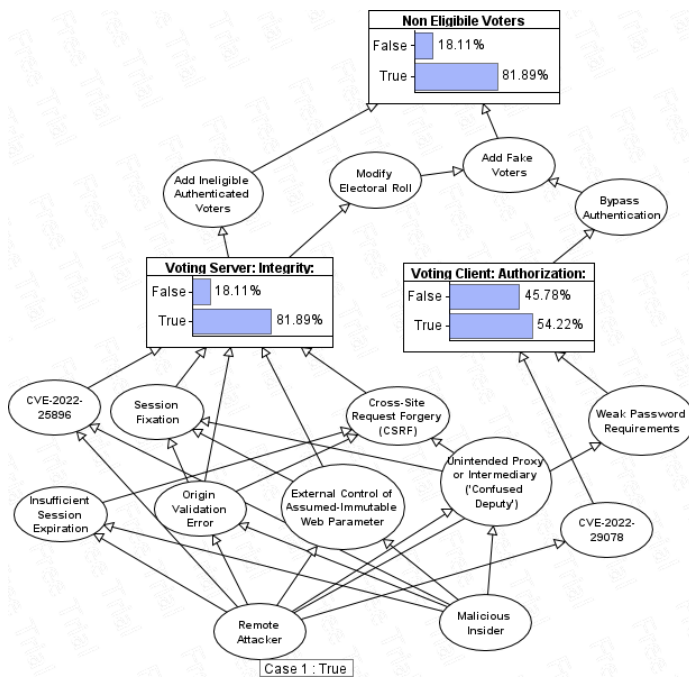


Fig 19. Calculation of the risk scenario when a remote attacker is trying to add ineligible voters.

## 6 Discussion

From the analysis of the generated attack graphs it could be observed that the biggest shortcoming in the *Chirotonia* is the level of responsibility that the proxy web server carries. Even though the protocol paper suggests that the web server is a proxy between a voting client and the blockchain with the sole purpose of anonymization potential (with multiple proxies deployed) the analysed system implementation puts the web server in the position of enforcing the voting protocol itself. This exposes the election to many known and unknown vulnerabilities that exist in the traditional web environment. Moreover, not enough protocol procedures are verified against the blockchain, the most important of which is the tally procedure. The tally calculation based on the ballot box maintained by the web server is a crucial weakness in the *Chirotonia* system. These observations supplement the results of the Voatz security analysis [SKW20] and arguments provided in the [PSN<sup>+</sup>21] which conclude that blockchain usage in the classical web environment increases probability of busting the ballot before it reaches the untamperable blockchain ledger.

Another weak point in the *Chirotonia* is the number of assets that the voting device has access to. End devices are traditionally considered to be the weakest elements in the overall system. Due to the high probability of end device compromise, and the lack of knowledge that the voter may hold, the system should be developed with assumption that the end device is compromised. However, *Chirotonia* explicitly imposes the safe storage of cryptographic and election assets on the voter. Moreover, the voting card could easily be sold after the registration phase. The system does not include any verification that the same device or the same user is participating in the registration and voting phases. National voting systems could use the keys available in the voter's identity cards thus mitigating cryptographic material's exposure. The addition of the voting card asset is doubtful. The performed analysis suggests that the fewer assets the election protocol requires the safer is the environment.

Verification procedure in *Chirotonia* is purely blockchain-based. This does not increase the security for several reasons: (i) as mentioned before, the main carrier of the protocol is the web server, for which user verification is non-existent; (ii) the blockchain transactions provided in the Ethereum explorer are not human-readable and does not provide any additional knowledge to the average user. In order to make the election receipt human readable it would have to be transformed by some intermediate component, most likely, a web server. This introduces the same issue of malicious web servers forging ballot receipts. Even though the blockchain-driven electronic voting visions offer public verifiability of the ballot box, in practice this is difficult to achieve.

Systems like Helios [Adi08] explicitly state that the election protocol execution is correct as long as the centralised web server is not compromised. It seems that the current implementation of *Chirotonia* requires the same assumption. This contradicts the promises made by the introduction of the blockchain.

## 7 Conclusion and Future Work

This work presents a framework to assess risk of a system implementation based on Bayesian attack graphs. Using the proposed framework, a blockchain-based electronic voting system is analysed and the weakest protocol and implementation aspects are identified.

It can be concluded that Bayesian attack graph driven risk assessment can aid election officials in making decisions about the suitability of electronic voting system architectures and implementations. Vulnerability scanners are constantly developed to automatically find software and configuration weaknesses, and the correct ontological organisation of the mined knowledge provides fast and readable insights in the form of probabilistic graphs. Dynamic analysis of each system state's impact on the overall system and terminal node states demonstrates the most crucial vulnerabilities and components. Analysing the concrete implementation vulnerabilities rather than formalised system models is a crucial step in evaluating its suitability. Software development contains details that may be missed in abstract modelling. Careful and comprehensive enumeration of knowledge is the principal activity in risk assessment. CVE, CWE and CAPEC databases proved to be sufficient for initial analysis. However, to analyse the system in more detail, the knowledge should be tailored to the specific environment, especially the probability calculations.

Tracking the legal compliance of the system is a difficult task. This work informally mapped legal regulations to the system threats which in turn were mapped to specific controls in the system. However, for more extensive legal compliance analysis, the threats should be elicited in a more formal way, potentially employing some language processing or models. The threat mappings to the system properties could be automated (e.g. with model checking) as well to reach more detailed analysis.



## References

- [Adi08] Ben Adida. Helios: web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium, SS'08*, pp. 335–348, San Jose, CA. USENIX Association, 2008.
- [Bou16] Philip Nicholas Boucher. What if blockchain technology revolutionised voting?, 2016.
- [BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC '94*, pp. 544–553, Montreal, Quebec, Canada. Association for Computing Machinery, 1994. ISBN: 0897916638. DOI: 10.1145/195058.195407. URL: <https://doi.org/10.1145/195058.195407>.
- [Cha04] D. Chaum. Secret-ballot receipts: true voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004. DOI: 10.1109/MSECP.2004.1264852.
- [Cla14] Mark Clayton. Ukraine election narrowly avoided 'wanton destruction' from hackers. 2014. URL: <https://www.csmonitor.com/World/Passcode/2014/0617/Ukraine-election-narrowly-avoided-wanton-destruction-from-hackers> (visited on 2023-04-27).
- [CLC17] Rommel N Carvalho, Kathryn B Laskey, and Paulo CG Costa. Pr-owl—a language for defining probabilistic ontologies. *International Journal of Approximate Reasoning*, 91:56–79, 2017.
- [CM16] Dylan Clarke and Tarvi Martens. E-voting in Estonia. *Real-World Electronic Voting: Design, Analysis and Deployment*:129–141, 2016.
- [Cou17] Committee of Ministers Council of Europe. Recommendation CM/Rec(2017)5[1] of the committee of ministers to member states on standards for e-voting. 2017. URL: [https://search.coe.int/cm/Pages/result\\_details.aspx?ObjectId=0900001680726f6f](https://search.coe.int/cm/Pages/result_details.aspx?ObjectId=0900001680726f6f) (visited on 2023-04-27).
- [CR20] Amrita Chatterjee and Hassan Reza. Toward modeling and verification of uncertainty in cyber-physical systems. In *2020 IEEE International Conference on Electro Information Technology (EIT)*, pp. 568–576, 2020. DOI: 10.1109/EIT48999.2020.9208273.
- [Eur18] Scientific Foresight Unit European Parliamentary Research Service. Prospects for e-democracy in europe. 2018. URL: [https://www.europarl.europa.eu/thinktank/en/document/EPRS\\_STU\(2018\)603213](https://www.europarl.europa.eu/thinktank/en/document/EPRS_STU(2018)603213) (visited on 2024-04-12).
- [GHL18] Ni Gao, Yiyue He, and Beilei Ling. Exploring attack graphs for security risk assessment: a probabilistic approach. *Wuhan University Journal of Natural Sciences*, 23(2):171–177, 2018.

- [HLL19] Wenhao He, Hongjiao Li, and Jinguo Li. Unknown vulnerability risk assessment based on directed graph models: a survey. *IEEE Access*, 7:168201–168225, 2019. DOI: 10.1109/ACCESS.2019.2954092.
- [HLP<sup>+</sup>20] Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 644–660, 2020. DOI: 10.1109/SP40000.2020.00048.
- [HMM<sup>+</sup>23] Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Ivan Pryvalov. Sok: secure e-voting with everlasting privacy. *Proceedings on Privacy Enhancing Technologies*, 1:279–293, 2023.
- [iEur14] Europos Parlamentas ir Europos Sąjungos Taryba. 2014 m. liepos 23 d. Europos Parlamento ir Tarybos reglamentas (es) nr. 910/2014 dėl elektroninės atpažinties ir elektroninių operacijų patikimumo užtikrinimo paslaugų vidaus rinkoje, kuriuo panaikinama Direktyva 1999/93/eb. 2014. URL: <https://eur-lex.europa.eu/legal-content/LT/TXT/?uri=CELEX%3A32014R0910> (visited on 2023-04-27).
- [Ins07] National Democratic Institute. Re-evaluation of the use of electronic voting in the netherlands. 2007. URL: <https://www.ndi.org/e-voting-guide/examples/re-evaluation-of-e-voting-netherlands> (visited on 2023-04-27).
- [Ins09] National Democratic Institute. The constitutionality of electronic voting in germany. 2009. URL: <https://www.ndi.org/e-voting-guide/examples/constitutionality-of-electronic-voting-germany> (visited on 2023-04-27).
- [JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. *Coercion-resistant electronic elections*. In *Towards Trustworthy Elections: New Directions in Electronic Voting*. David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Mirosław Kutylowski, and Ben Adida, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 37–63. ISBN: 978-3-642-12980-3. DOI: 10.1007/978-3-642-12980-3\_2. URL: [https://doi.org/10.1007/978-3-642-12980-3\\_2](https://doi.org/10.1007/978-3-642-12980-3_2).
- [JRS<sup>+</sup>04] David Jefferson, Aviel D Rubin, Barbara Simons, and David Wagner. A security analysis of the secure electronic registration and voting experiment (SERVE). *Recuperado de: http://euro.ecom.cmu.edu/program/courses/tcr17-803/MinorityPaper.pdf*, 2004.
- [LM05] Yu Liu and Hong Man. Network vulnerability assessment using bayesian networks. In *Data mining, intrusion detection, information assurance, and data networks security 2005*, vol. 5812, pp. 61–71. SPIE, 2005.
- [LWW04] Joseph K. Liu, Victor K.-W. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). *IACR Cryptol. ePrint Arch.*, 2004:27, 2004.

- [Min18] Lietuvos Respublikos Teisingumo Ministerija. Lietuvos Respublikos balsavimo internetu pagrindų įstatymo projektas. 2018. URL: <https://e-seimas.lrs.lt/portal/legalAct/lt/TAP/e5d36ad00d9b11e88a05839ea3846d8e?jfwid=ozvxl44vs> (visited on 2023-04-27).
- [MS02] Klaus Mosegaard and Malcolm Sambridge. Monte carlo analysis of inverse problems. *Inverse problems*, 18(3):R29, 2002.
- [MSB<sup>+</sup>17] Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil C Lupu. Exact inference techniques for the analysis of bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 16(2):231–244, 2017.
- [MVT<sup>+</sup>19] Salman Manzoor, Tsvetoslava Vateva-Gurova, Rubén Trapero, and Neeraj Suri. Threat modeling the cloud: an ontology based approach. In *Information and Operational Technology Security Systems: First International Workshop, IOSec 2018, CIPSEC Project, Heraklion, Crete, Greece, September 13, 2018, Revised Selected Papers 1*, pp. 61–72. Springer, 2019.
- [Neu16] Stephan Neumann. *Evaluation and improvement of internet voting schemes based on legally-founded security requirements*. PhD thesis, Technische Universität Darmstadt, 2016.
- [NM<sup>+</sup>01] Natalya F Noy, Deborah L McGuinness, et al. *Ontology development 101: a guide to creating your first ontology*, 2001.
- [OGA<sup>+</sup>05] Xinming Ou, Sudhakar Govindavajhala, Andrew W Appel, et al. Mulval: a logic-based network security analyzer. In *USENIX security symposium*, vol. 8, pp. 113–128. Baltimore, MD, 2005.
- [PDR11] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2011.
- [Pis21] Anna Piscitelli. *Sharing responsibilities through Distributed Key Generation in the Chironia e-voting framework: a blockchain-based approach*. MA thesis, University of Naples, Federico II, 2021.
- [PJS21] Adrien Petitpas, Julien M. Jaquet, and Pascal Sciarini. Does e-voting matter for turnout, and to whom? *Electoral Studies*, 71:102245, 2021. ISSN: 0261-3794. DOI: <https://doi.org/10.1016/j.electstud.2020.102245>. URL: <https://www.sciencedirect.com/science/article/pii/S0261379420301244>.
- [PSN<sup>+</sup>21] Sunoo Park, Michael Specter, Neha Narula, and Ronald L Rivest. Going from bad to worse: from Internet voting to blockchain voting. *Journal of Cybersecurity*, 7(1), 2021-02. ISSN: 2057-2085. DOI: 10.1093/cybsec/tyaa025. eprint: <https://academic.oup.com/cybersecurity/article-pdf/7/1/tyaa025/42533672/tyaa025.pdf>. URL: <https://doi.org/10.1093/cybsec/tyaa025>. tyaa025.

- [RAV<sup>+</sup>21] Antonio Russo, Antonio Fernández Anta, Maria Isabel González Vasco, and Simon Pietro Romano. Chirotonia: a scalable and secure e-voting framework based on blockchains and linkable ring signatures, 2021. URL: <https://arxiv.org/abs/2111.02257>.
- [rkom21] Vyriausioji rinkimų komisija. Sprendimas dėl internetinio balsavimo informacinės sistemos galimybių studijos. 2021. URL: <https://e-seimas.lrs.lt/portal/legalAct/lt/TAD/0f9a17225e6411ecb2fe9975f8a9e52e> (visited on 2023-04-27).
- [RRI16] Peter YA Ryan, Peter B Rønne, and Vincenzo Iovino. Selene: voting with transparent verifiability and coercion-mitigation. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, pp. 176–192. Springer, 2016.
- [SFD<sup>+</sup>14] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J Alex Halderman. Security analysis of the Estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 703–715, 2014.
- [SH]<sup>+</sup>02] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pp. 273–284. IEEE, 2002.
- [SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – EUROCRYPT ’95*, pp. 393–403, Berlin, Heidelberg. Springer Berlin Heidelberg, 1995. ISBN: 978-3-540-49264-1.
- [SKW20] Michael A Specter, James Koppel, and Daniel Weitzner. The ballot is busted before the blockchain: a security analysis of voatz, the first internet voting application used in us federal elections. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pp. 1535–1552, 2020.
- [SOO07] Adesina S Sodiya, S Adebukola Onashoga, and BA Oladunjoye. Threat modeling using fuzzy logic paradigm. *Informing Science: International Journal of an Emerging Transdiscipline*, 4(1):53–61, 2007.
- [TT20] Ruhi Taş and Ömer Özgür Tanrıöver. A systematic review of challenges and opportunities of blockchain for e-voting. *Symmetry*, 12(8), 2020. ISSN: 2073-8994. DOI: 10.3390/sym12081328. URL: <https://www.mdpi.com/2073-8994/12/8/1328>.

- [WHR<sup>+</sup>03] Warren E Walker, Poul Harremoës, Jan Rotmans, Jeroen P Van Der Sluijs, Marjolein BA Van Asselt, Peter Janssen, and Martin P Kraye von Krauss. Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support. *Integrated assessment*, 4(1):5–17, 2003.
- [Wil22] Nathaniel Williams. Remote Voting in the Age of Cryptography. *MIT Computational Law Report*, 2022-12. <https://law.mit.edu/pub/remotevotingintheageofcryptography>.