From Centralized to Decentralized Remote Electronic Voting

Christian Killer, Bruno Rodrigues, Eder John Scheid, Muriel Figueredo Franco, and Burkhard Stiller

1 Introduction

Electronic Voting (EV) is straightforward in an ideal and secure world: voters send their votes to a trusted counter, which tallies the votes and publishes the final result. However, that is far from being realistic when the opposite is the case: entities involved in EV are typically not fully transparent and trusted, and the context is adversarial. In a traditional EV setting, every participant in an election system is potentially malicious, for example, voters selling votes, dishonest election officials, and corrupt equipment vendors [42].

Therefore, opportunities exist to decentralize electoral process stages and promote transparency to reduce dependency on centralized entities. While decentralization reduces the control of entities that may be directly interested in the election result, the challenge lies in guaranteeing verifiability and privacy of all processes. This challenge directly addresses trust assumptions that a centralized system explicitly places on a single entity for (i) the execution of the voting protocol, (ii) the verifiability and privacy properties, and (iii) the audit trail's storage. Moreover, voting systems' essential privacy requirements oppose the need for convincing evidence that the election's outcome is correct while assuring ballot secrecy at the same time [6].

Voting research is interdisciplinary and spans decades and many fields, ranging from technical concepts, such as network security and cryptography, to socioeconomic aspects (e.g., statistics and law). Over the years, advances in research allowed for the deployment of EV systems, previously only formulated in theory.

C. Killer · B. Rodrigues · E. J. Scheid · M. F. Franco · B. Stiller (✉)
Department of Informatics IfI, Communication Systems Group CSG, Universität Zürich UZH, Zürich, Switzerland
e-mail: killer@ifi.uzh.ch; rodrigues@ifi.uzh.ch; scheid@ifi.uzh.ch; franco@ifi.uzh.ch; stiller@ifi.uzh.ch

S. Ruj et al. (eds.), *Blockchains*, Advances in Information Security 105,
https://doi.org/10.1007/978-3-031-32146-7_16

For example, [54] categorizes existing EV approaches into (i) paper-based processes with electronic counting and (ii) digital systems that use direct recording systems (i.e., an electronic ballot box) and remote voting either in a dedicated network or remotely over the Internet. Advances in these different approaches enabled the practical evaluation of cryptographic schemes in real-world scenarios. Poll site systems (i.e., systems where voters are required to vote in a predetermined voting location with an electronic ballot) were extended with EV to provide a verifiable audit trail based on cryptographic voting schemes [2, 8, 57].

As a special case in EV approaches, Remote EV (REV) systems were designed and implemented to allow for vote casting in a remote and uncontrolled environment (e.g., from a personal device, typically today over the Internet). For that reason, REV is often referred to as "Internet Voting" (or I-Voting). Depending on the specific legal and country-specific context, REV and I-Voting may be considered synonymous. So far, various electoral processes used REV systems in the last few years, e.g., Switzerland [76], Estonia [43, 83], Norway [34], and Australia [41]. However, these systems are in major revision processes today due to security concerns raised, resulting in REV attempts being put on hold. Therefore, a secure implementation and deployment of a scalable and operational REV system still requires utmost scrutiny of every aspect of the system itself, its stakeholders involved, and its deployment and ultimately requires trust from the electorate.

All REV systems require Public Bulletin Boards (PBB), which are functionally comparable to final tally boards of traditional voting approaches and immutably store ballots to provide a verifiable audit trail to the public. In state-of-the-art REV implementations, PBBs (which do determine a central component for any REV system) are formulated as "trusted centralized components" and are assumed to provide the verifiable audit trail [1, 14, 19]. As further prior work outlined, the use of permissioned Blockchains (BC) or permissioned Distributed Ledgers (DL) provides an immutable, publicly shared, and append-only PBB [49, 51]. To reach these requirements, N trusted authorities determine a permissioned DL-network, formed by a Proof-of-Authority (PoA) consensus, which measurably lowers trust assumptions from BFT (Byzantine Fault Tolerance)-based networks from $\frac{2}{3}$ to at least $\frac{N}{2} + 1$ honest nodes. The BC's and DL's underlying data structures in place today provide favorable properties for audit trails, and hence, for PBBs. Thus, because of BC's and DL's immutable, append-only, and publicly readable structure, they were used to implement voting systems in related [15, 59] and prior work [48–50].

To establish an objective basis for understanding such distributed approaches, it is fundamental to differentiate between *permissionless* and *permissioned* decentralized networks. By definition, permissionless consensus algorithms, e.g., Proof-of-Work (PoW) or Proof-of-Stake (PoS), are based on an economic incentivization of the honest behavior of the majority of participants while punishing malicious behavior, which can always be detected. The major goal of permissionless networks is to remove the need for Trusted Third Parties (TTP) (e.g., as known for financial transactions without intermediaries), thus, any intermediary. Through logical

deduction, permissionless consensus algorithms are determined to be unsuitable for those use cases that already assume a TTP inherently. For instance, voting and election systems for governments and organizations are, by definition, led by trusted authorities (at least this can be safely assumed for a number of societies across the globe). Therefore, permissioned networks offer a highly suitable framework for use cases, where a set of N trusted authorities collaborate in their roles as defined by law or identity as determined by society without any economic incentive. Unfortunately, so far, the full set of implications (e.g., long-term privacy) are not yet considered when selecting a permissionless BC.

REV Core Challenges With the steady decline of trust in governments and elections [22, 80], it is imperative to address public service provisioning in a decentralized way [71]. In political sciences, efforts toward Electronic Government (E-Government) reverse such declines in trust. Still, whether this has a measurable effect on trust or whether it affects the government's perceived efficiency is disputed [80]. From a technological perspective, security-sensitive software is often not open-sourced [84], because of various interests colliding (e.g., commercial and national security interests). Although it is not guaranteed that open-source software is "automatically" more secure [75], voting systems require certainly and without restrictions always transparency and public scrutiny in order to build trust among the general public.

When it comes to fostering trust in electoral processes, the lack of transparency is very often the major concern. Without transparency, false accusations cannot be debunked, which undermines the trust in the electoral process. This problem can be tackled by publishing all verifiable data on a PBB, which is immutable and publicly readable. BCs and DLs highlight a feasible possibility to minimize or distribute trust among stakeholders. In contrast, in traditional REV systems, voters (i.e., the electorate) and Voting Authorities (VA) are the primary stakeholders. The public trust in the voting process (and its outcome) is the foundation of democracy, and it is based on verifiable evidence, which voters can publicly query. While VAs manage voting procedures, voters expect verifiable proof to conduct audits of these procedures.

Although there are many open challenges and threats in REV systems that will not be solved or mitigated with the usage of decentralized systems, there are two core challenges that REV systems can address in full and proven reliably: (i) achieving a tamper-proof and immutable audit trail in a practical PBB deployment and (ii) achieving a practical decentralization of trust by deploying a suitable consensus algorithm (e.g., Proof-of-Authority, PoA). While many research papers and industry efforts proposed BC- and DL-based REV systems, only very few of them thoroughly considered the detailed trust assumptions required and trade-offs to be made explicit, which need to be evaluated upon deploying a decentralized system in a highly adversarial context [61].

Contributions While both EV and REV systems expose trade-offs between election verifiability and privacy, REV systems add even higher complexity, since they

do not run within a controlled environment, i.e., outside of a prepared voting booth, where eligible voters cast their votes. In this regard, REV's fundamental goal is to achieve privacy while providing a fully verifiable audit trail. However, many of the BC- and DL-based systems proposed did not consider such significant requirements for REV systems, e.g., the trust model and assumptions of the underlying consensus mechanisms. Thus, transparency is a key factor for trust in REV systems. *For example*, the Swiss ordinance regarding REV requires any potential REV system being introduced to publish the source code for the public without exceptions [25].

Therefore, this chapter's major contribution is to address Remote Electronic Voting (REV) in exactly that context and to discuss trade-off settings for clarifying how Distributed Ledgers (DL) can enhance the decentralization of the electoral process by ensuring transparency and the ability to verify results while guaranteeing voter confidentiality at the same time.

Organization This chapter is organized as follows. Section 2 provides major background concepts for REV, such as Public Bulletin Boards (PBB) and protocol requirements, including their associated privacy properties. In a similar direction, Sect. 3 describes cryptographic primitives and underlying assumptions that are essential for REV and PBBs. Section 4 overviews the REV state of the art by discussing how cryptographic elements and mechanisms are leveraged to fulfill these previously outlined protocol requirements. Additionally, Sect. 5 outlines research directions and open aspects for REV systems considering as core principle a privacy-preserving requirement as the principal design driver for both centralized and decentralized REV implementations. Finally, Sect. 6 provides overall considerations based on those significant findings outlined and addressed herein.

2 Remote Electronic Voting (REV) Fundamentals

The core challenge in REV is to balance privacy and verifiability properties based on a transparent and evidence-based voting process. Many early REV systems used trusted central servers as a PBB, which represents a single point of failure operated by trusted stakeholders that may have an interest in the election's outcome [38]. For example, the Swiss Post (SP) Voting system operates in trusted, closed, and hardened datacenters under the control of the SP (a partially state-owned company). Although the protocol assumes only a set of separate control components as trustworthy, they are all technically operated by the SP [39]. The overall system is also dependent on a code-verification system, which assumes a trustworthy printing office that secretly prints and sends code sheets via postal mail to eligible voters. In the case of collusion between the printing office and compromised devices, the device can return correct verification codes, although votes are not cast as intended by the voter [6].

Ideally, a cryptographic voting scheme would allow voters to only trust cryptographic schemes. It would not be required to assume external stakeholders'

trustworthiness (e.g., a print office's printing verification code). In order to minimize these trust assumptions placed in centralized infrastructure, Blockchains (BC) and Distributed Ledgers (DL) are a suitable possibility to (i) minimize and distribute trust among stakeholders and (ii) enable practical implementation of an immutable and append-only PBB for REV systems. However, many of the BC- and DL-based systems proposed did not consider the complete set of requirements for REV systems, e.g., a trust model and assumptions of underlying consensus mechanisms. Persisting election evidence on a BC or DL enhances trust assumptions marginally, mainly because the election evidence's immutability is assured. Further, the practical decentralization of the voting process on an architectural layer can be achieved as well.

> We need to realize popular services in a secure, distributed, and decentralized way, powered by free software and free/open hardware. We need to build systems beyond the reach of super-sized companies and spy agencies. Such services must be based on strong cryptography. (Phillip Rogaway [71])

The design goals are similar and reasonable: increase transparency and provide a publicly readable and fully decentralized audit trail. After all, transparency is a critical factor for trust in REV systems. For instance, the Swiss Federal Ordinance for Federal Chancellery Ordinance on Electronic Voting (VEleS) [25] demands the publication of source code for any REV system that is used to tally more than 30% of the cantonal electorate. In order to assess the feasibility of any system, the architecture and its components need to be assessed. Therefore, the major building blocks of an REV system are key, with the core component of a decentralized REV system being built by the Public Bulletin Board (PBB). Voters use a Voting Client that executes a Voting Software Client (VSC) to interact with the PBB directly through a set of defined communication channels while interacting according to the voting protocol to ensure privacy and verifiability.

2.1 Building Blocks of an REV Architecture

Before detailing the key privacy and verifiability properties of an REV system, Fig. 1 depicts the main building blocks needed to design an REV system. In essence, the *Voter* uses a *Voting Client* to execute *Voting Software*, which was distributed by the *Voting Authorities (VA)*. A voter uses personal hardware capable of executing the voting software, e.g., a smartphone or personal computer. The voter fetches the official voting software from publicly available storage (e.g., a Web site, repository, or maybe a distributed storage). The integrity and authenticity of the voting software executable must be assured (i.e., cryptographically signed with the VA's credentials). Depending on the voting protocol, the *Ballot* is generated, which can include various cryptographic operations that need to be executed on the voting client. For instance, certain Zero-Knowledge Proofs (ZKP) may be required to prove ballot validity. A crucial component of REV systems is *Public Bulletin Boards (PBB)*, which are part

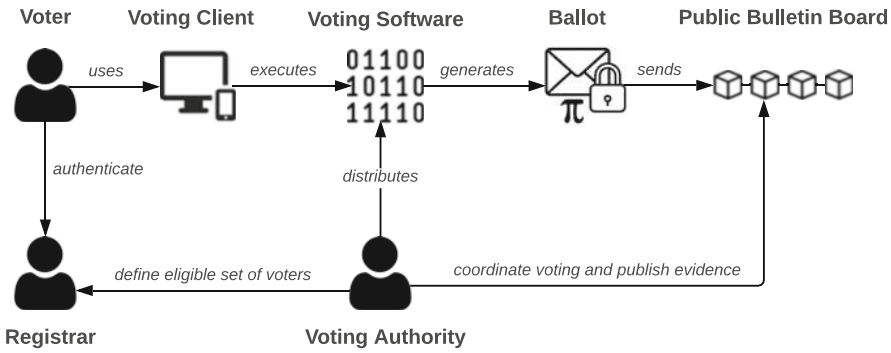

Fig. 1 Overview of major REV building blocks

of any REV system, used to publish an auditable evidence trail during the execution of the voting scheme [49]. According to [44, 45], a PBB shall provide the following properties:

1. It is an append-only data structure, i.e., information cannot be modified or altered.
2. It is public in the sense of being searchable by anyone.
3. It is consistent in its view for anyone accessing its information.

Voting systems require verifiable evidence of a correctly executed voting process. In that context, the PBB is the most relevant component of an REV system that a BC or DL can replace since it requires similar properties: immutability, be append-only, and persist a consistent view to anyone. Voters can query the PBB at any time to verify evidence published by the VA.

Another crucial aspect of REV systems is voter authentication i.e., assuring that only eligible voters can cast a ballot. Often, REV systems use a trusted *Registrar* entity to authenticate and check the eligibility of voters and provide them with the necessary credentials to vote. Hence, Identity Management (IdM) is crucial to give credentials to eligible voters. Usually, the IdM is handled by a TTP, such as the Voting Authorities (VA). The IdM manages voters' identities and provides credentials to the eligible set of voters, which can be used to authorize access to cast valid ballots. It is important that the IdM cannot fake identities because it would enable ballot-stuffing.

Another crucial aspect is the security of the communication channels used between the different entities. These channels need to be secured in order to guarantee various security properties, as well as the integrity of the full voting process. For instance, *sender-anonymous* channels and *untappable communication* channels are required [44]. *Sender-anonymous* channels ensure that a sender cannot be identified by the receiver of a message, whereas *untappable communication* channels ensure that neither the sender nor the receiver "can learn anything about the communication, including whether communication occurred or not" [44]. Depending on the specific voting protocol used, these assumptions need to be

carefully assessed with the practical deployment decisions, e.g., Over which channel does the voting software interact with any of the other components?

2.2 Architectural Models for Public Bulletin Boards

The use of a platform based on DLs for the construction of elections based on REV does not necessarily imply the use of BCs. It is fundamental to note the differences between the architectural models used in the different types of storage architectures (cf. Fig. 2). For instance, a permissionless BC-based architecture implies open participation in the consensus mechanism, i.e., every participant running a full node is able to participate in the consensus process. While traditional permissioned and centralized database storage may result in a lack of transparency in the electoral process, decentralized options may also have drawbacks, such as loss of efficiency and greater overhead in managing the electoral process stages.

Considering Fig. 2 and the distinction between centralized and distributed architectures for decentralized architectures, both are managed by only one organization and can count on one server (or a pool of servers in a datacenter) in a centralized architecture or several servers in a distributed architecture. In these two types of architectures, it is necessary to trust that the entity (i.e., federation) controlling the electoral process is not being influenced by adversaries. Thus, the transparency of the process is a fundamental element that an organization does not always guarantee but can be more easily obtained through N independent entities in a decentralized and transparent architecture.

Therefore, it is crucial to distinguish between BC and DL technologies through which it is feasible to decentralize electoral process where each participating entity is independent, (i.e., the decentralized process is federalized). In a decentralized REV, each entity is independent and can publish its results on the PBB without

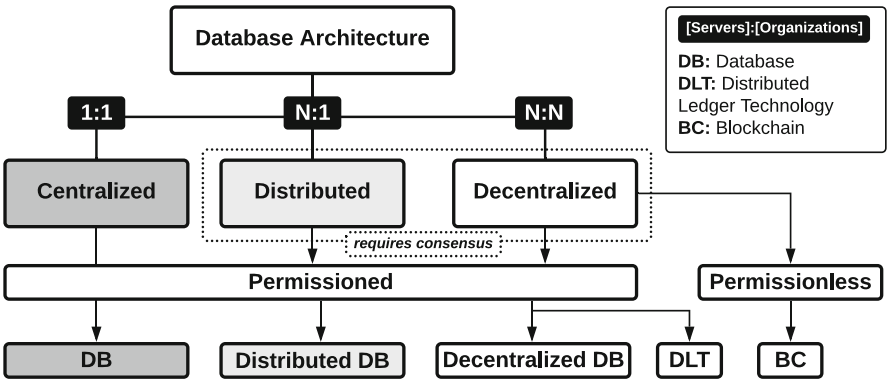

Fig. 2 Taxonomy of data structures and consensus algorithms

relying on another entity. This is a relatively more transparent process than a centralized REV.

2.3 Protocol Requirements

Literature on EV offers various protocols to meet the core requirements of privacy and verifiability, which are in direct opposition to one another and inherently contradictory, e.g., Individual Verifiability (IV) seems to be incompatible with Receipt-Freeness (RF) [36]. Also, [18] has shown that a voting system cannot achieve Unconditional Privacy (UP) and Universal Verifiability (UV) at the same time. They also showed that UV and RF could not coexist if the transcript of the vote depends on *only* public values and values known to the voter. For this reason, REV systems aim to achieve a balance between privacy and verifiability.

2.3.1 Privacy Properties

Vote Privacy is considered a human right and is stated as such in Article 21 of the Universal Declaration of Human Rights [82]. A voting system without Ballot Privacy allows large-scale vote-buying and voter coercion [73]. Thus, privacy is key in democratic voting. In this context, privacy can be further classified in subcategories, which are based on the degrees of privacy and assumptions made, ranging from the weakest notion of Ballot Privacy (BP) to the strongest of Coercion-Resistance [24] (Fig. 3).

Many privacy properties require at least a minimal amount of trust placed in the Voting Authorities (VA) to store the private keys for the election keys safely. Trust in single authorities can be distributed by using threshold cryptography, hence distributing the trust among VAs and other trusted stakeholders. For instance, by using a BC- or DL-based system, it is possible to decentralize the point of trust in a single party through transparent infrastructure in which all participants can

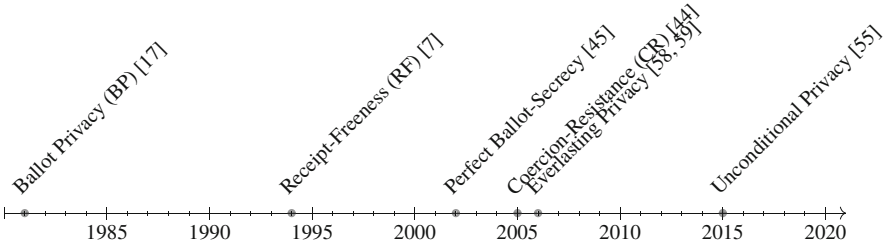

Fig. 3 Timeline of privacy notions

verify the distributed key generation, the confirmation of votes cast, and the vote tallying procedures.

> The will of the people shall be the basis of the authority of government; this will shall be expressed in periodic and genuine elections which shall be by universal and equal suffrage and shall be held by secret vote or by equivalent free voting procedures. (Article 21, (3) of Universal Human Rights Declaration [82])

In 1981, the first notion of *Ballot Privacy (BP)* was defined by [17] as "A voter's vote is not revealed to anyone." In [20], the notion was formally introduced, laying the groundwork for formal notions of voting system properties. Often, the term *Ballot Secrecy* is used interchangeably to describe BP. The National Institute of Standards and Technology (NIST) defined Ballot Secrecy in a more process-oriented fashion as follows: "a voting system is private if it (and the procedures/process for using it) does not make available additional information on an individual voter's ballot choice(s), beyond that contained in the tally" [58]. Later, in 2002, another notable definition followed from [47], called *Perfect Ballot Secrecy*, which "ensures that knowledge about the partial tally of the ballots of any set of voters is only computable by the coalition of all the remaining voters (this property captures strong voter privacy as understood in real world elections)." Further refinements of this definition can be found in literature based on the trust model or the computational power of an adversary [47, 56, 60].

Most early works on voting systems had underlying protocols that produced receipts. These receipts would allow voters to prove how they voted to others. Intuitively, the ability to produce a receipt of how a voter voted can easily be abused for vote-buying or coercion [7]. This brought about the definition of *Receipt-Freeness (RF)*, which is defined as "a voter cannot gain information which can be used to prove, to a vote-buyer, how she voted" [7].

The key difference between RF and *Coercion-Resistance (CR)* is whether the voter is considered malicious or not. In the case of RF, the voter is considered to be a dishonest voter who wishes to sell her vote. For CR, it does not matter whether the voter is honest or not. The voter is coerced by an adversary to disclose sensitive information, such as her vote or voting credentials. As such, CR is a stronger privacy guarantee than RF. Therefore, CR is defined as "a voter cannot interact with a coercer to gain information, which can be used to prove how she voted" [44].

The privacy definitions above do not explicitly define an adversarial model, i.e., do not include any assumptions regarding the adversary's computational capabilities. While current adversaries are bound by hardware and software available as of now, future adversaries may break underlying assumptions, which in turn break the security of cryptographic primitives and compromise the privacy properties. Many voting schemes are based on these hardness assumptions, such as the Decisional Diffie-Hellman assumption (DDH) [11], which is covered in detail in Sect. 3.3. With a focus on these intractability assumptions, further definitions of *everlasting* and *unconditional* BP notions are reviewed.

A voting system with *Everlasting BP (EP)* does not allow even a computationally unbounded coalition of voters or outside observers to determine for whom a voter

voted [60]. Going even further is *Unconditional BP (UP)*. A voting system with UP does not allow anyone, neither voters, outside observers, nor VAs, be they computationally unbounded or not, to determine for whom another voter voted [56].

Remembering the definition of perfect BP [47], even if VAs conspire, they are unable to link votes to voters and break BP. Therefore, a voting scheme with perfect BP does not require trust in the VAs or other parties. However, perfect BP does not mention any computational hardness assumptions and, hence, is not everlasting. Therefore, by combining everlasting BP (i.e., no computational hardness assumptions for privacy) and perfect BP (i.e., no trust in TTPs needed for BP), it is possible to achieve the Unconditional BP.

2.3.2 Verifiability

In real-world systems, voters have to place implicit trust in voting machines and polling staff for the integrity of the elections. In the pursuit of minimizing the requirement for trusted personnel and hardware, REV research proposed cryptographically verifiable voting schemes [3].

Once all votes have been cast, the system must be able to prove that the result was tallied correctly and that the single ballots contain the actual voters' choices. Prior work proposed many different notions of verifiability, and novel, formal definitions are being researched as well. Sako and Kilian [74] defined *Individual Verifiability (IV)* as "a voter can verify that her vote is included in the set of all votes cast."

However, IV is not sufficient to provide a verifiable REV system because the system then relies on each voter to verify that each ballot was correctly published and unaltered. Since that is an unrealistic assumption, [74] also defined Universal Verifiability (UV) as being achieved if "an observer can verify that the tally has been correctly computed from the set of all votes cast" [74].

Additional informal definitions of UV by Kremer et al. [53] are defined as being achieved if "anyone can verify that any vote in the set of "all" votes belongs to an eligible voter, and each eligible voter voted only once [53]." This notion already includes eligibility verifiability, highlighting the challenges of ensuring that only eligible voters vote and only vote once.

Further research in the realm of verifiability properties refined a more procedural perspective on the *end-to-end* voting process, called *End-to-End Verifiability (E2E-V)*, which is only fulfilled if a voter can verify the following three properties [44]:

- *Cast-as-Intended* (CaI): "her choice was correctly denoted on the ballot by the system."
- *Recorded-as-Cast* (RaC): "her ballot was received the way she cast it."
- *Counted-as-Recorded* (CaR): "her ballot counts as received."

The last property of E2E-V, CaR, is often also referred to as *Tallied-as-Recorded* (TaR) as well. Therefore, voters first confirm to the system that the vote has been correctly encrypted, and thus CaI is achieved (by verifying the encryption operation

on a second device, for instance). Then, the voter tracks her vote on the PBB by using her receipt and can verify that the vote was RaC. Lastly, the result's integrity is ensured by auditing the tallying process, which means that the REV system publishes cryptographic proofs that prove correct decryption and operation [3].

However, these verifiability properties rely on a PBB as an audit trail; otherwise, there is no data to verify. A PBB serves as an immutable, append-only audit trail. Therefore, verifiability properties *assume* such a PBB exists. In practice, a fully decentralized REV system requires a shared PBB. A public permissioned DL can serve as such and enable these different verifiability properties, allowing voters, auditors, and the general public to query the persisted data. It is crucial that the deployment of said BC or DL is carefully parameterized and configured to the use-case and voting protocol. For instance, there is a huge difference between simply persisting hashes on a public permissionless BC and having actual VAs perform Multi-Party Computation within a permissioned DL network. These differences make it necessary to look at the properties within the specific context and mode of deployment of the PBB.

Küsters and Müller [55] drove efforts forward to elevate the understanding of dependencies between various privacy and verifiability properties in their formal and informal definitions, as well as misconceptions and misunderstandings of these properties.

1. For instance, it is commonly believed that IV and UV are the same as E2E-V. However, IV and UV are neither sufficient nor necessary to achieve E2E-V.
2. Further, E2E-V alone is typically insufficient for practical purposes. REV systems should really be designed with accountability in mind.
3. Further, CR does not imply general privacy, but increasing the level of privacy can lead to a lower level of coercion-resistance.

These confusions and misconceptions highlight the importance of clear definitions and a clear assessment of these definitions. The privacy and verifiability notions are not fixed but are ever-evolving into different formalized approaches. An overview is provided in [55]. Also, privacy and verifiability are not the *only* protocol requirements. Two notable properties include the notions of Software Independence and Accountability in the context of REV systems.

Additionally, the requirement *Software Independence* can claim for a voting system *if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome* [70]. REVs are complex hardware/software systems. Minor errors or manipulations can lead to unpredictable errors, which adversaries might exploit. As such, Software Independence follows the approach of: *Verify the election results, not the voting system*.

Furthermore, *Accountability* is a more robust, extended form of verifiability, i.e., if the final tally does not correspond to how the voters actually voted, then VAs can be held accountable for the misbehavior. The misbehavior can be intentional or unintentional. Accountability is important for multiple practical reasons because it intensifies the incentive all parties have to follow their roles because they can be singled out in cases of misbehaviour [55].

2.4 General System Requirements

While these protocol requirements are essential for the formulation of voting protocols, the REV system also needs to provide multiple facets, many of which are not specific to REV systems but to information systems in general. These general security system requirements are less specific to REV, but still fundamental for REV in practice, as REV follows a clear software architecture. These general requirements may contradict protocol requirements [36]. For instance, software architecture characteristics usually refer to the *"-ilities"* that the system must provide: availability, reliability, testability, scalability, and deployability [68]. Also, besides these general requirements, convenience, flexibility, and usability are also desirable properties. Further requirements that address political, administrative, or even juridical questions are also crucial for REV in practice.

In the context of BC- or DL-based REV systems, the most crucial aspect is the definition of the security goals and the threat model because these build the basis for the system's risk assessment. For instance, when using a public permissionless BC as a basis, the system's architecture depends on the assumption that the public network is reliable, available, and scalable. Public permissionless networks are, by design, not controlled by known stakeholders. Therefore, it is hard to challenge or prove the properties and assumptions made. However, if a voting system relies on a known set of permissioned nodes, i.e., a public permissioned DL, then the assumptions, properties, and performance can be measured more precisely while still providing the benefits of transparency.

3 Cryptographic Primitives and Assumptions

This section focuses on the fundamental cryptographic primitives and their underlying assumptions. The following background is necessary to understand REV protocol approaches; how they work technically, i.e., how the underlying cryptographic mechanisms support the voting scheme itself; and based on which significant assumptions these primitives hold their security notions. Table 1 shows the most relevant cryptographic primitive schemes to understand REV systems and provides a brief description of each of them. Further, the most important cryptographic primitives are introduced, accompanied by the most essential *computationally* hard problems. Also, ElGamal message encodings are detailed in Sect. 3.1 because many REV protocols and systems are based on the ElGamal public-key cryptosystem [28]. This section outlines the most fundamental cryptographic primitives used in voting schemes and relates for an in-depth view of selected topics (e.g., hash functions) to the respective general security-related literature.

Chaum [16] first proposed Blind Signatures (BS) to enable untraceable electronic mail and payments, and BS find application in early REV schemes as well, which would work as follows: Suppose a voter v encloses her $ballot_v$ which is written on

Table 1 Overview of cryptographic primitives

Scheme	Description
Blind Signatures (BS)	Digital signature scheme to sign blinded messages
Homomorphic Encryption (HE)	Allows operations on ciphertext, manipulating the plaintext
Zero-Knowledge-Proofs (ZKP)	Prove knowledge of v interactively, without revealing v
Non-Interactive ZKP (NIZKP)	Prove knowledge of v non-interactively, without revealing v
Multi-party Computation MPC	Allows for removal of TTPs from distributed computation
Distributed Key Gen. (DKG)	Allows multiple parties to collaboratively generate a public key
Fiat-Shamir Heuristic (FS)	Technique to render ZKPs non-interactive using hash functions
Commitment Schemes (CS)	Commit to value v now and reveal commitment and v later
Mix Nets (MN)	Mix ciphertext to remove relation to previous ciphertexts

carbon paper, within an envelope, and seals it. The sealed envelope is given to a trusted registrar r. The registrar verifies (i) the identity and eligibility of v and (ii) verifies that v has not cast a ballot yet. Then, r signs the envelope, which includes the ballot on the carbon paper, and returns it to v. Now, the voter v can *unblind* the ballot and cast the signed carbon paper $ballot_v$ separately. In short, the use of BS [16] allows a registrar r to sign a message m without knowing its contents [44].

Another important cryptographic primitive is Homomorphic Encryption (HE). HE enables the use of encrypted data in such a way that its decrypted result is the same as if the operation was performed on the original plaintext data. Confidential data remains private while still being processed in an untrusted environment (e.g., a public cloud provider) [4, 31]. While fully HE is considered the holy grail of modern cryptography, partial HE is also useful for specific use cases, such as voting. HE is based on a homomorphism (i.e., a structure-preserving mapping) existing between plaintext and ciphertext operation, which can be defined with equality as follows:

$$\mathsf{Enc}_{\mathsf{pk}}(m_1) \otimes \mathsf{Enc}_{\mathsf{pk}}(m_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1 \oplus m_2)$$

HE allows for combining two encrypted messages into one without the use of pk. The operators $\oplus$ and $\otimes$ represent arbitrary operations that depend on the implementation of the cryptosystem. This homomorphic property can be used to tally votes in a voting system. Every voter V can encrypt his or her ballot $ballot_V$ using the public election key pk, resulting in the encrypted vote: $e_{\mathsf{pk}}(ballot_V)$ [44].

Suitable cryptosystems that provide a homomorphism partially under a single operation are ElGamal [28] (cf. Sect. 3.1), the Paillier cryptosystem [64], and RSA [69]. However, ElGamal and Paillier are the most popular choice for REV schemes because both systems are non-deterministic, which is one reason why RSA, although offering homomorphic properties, cannot be used for REV systems. A non-deterministic cryptosystem can provide the security notion of IND-CPA, i.e.,

IND-CPA combines the security goal of indistinguishable ciphertexts in combination with the chosen-plaintext attacker-model. IND-CPA is the most important security notion in applied cryptography and is referred to as *semantic security*. IND-CPA captures the notion that ciphertexts should not leak any information about plaintexts as long as the key is kept secret. Hence, one way to achieve IND-CPA is to use *randomized encryption*, which implies that encryption is *probabilistic*, while decryption remains deterministic [5].

While HE enables the confidentiality of the ballots, the validity of the ballots needs to be ensured. In other words: how can voters prove that the ballot is a valid ballot and not an invalid ciphertext? A suitable tool is *Zero-Knowledge Proofs (*ZKP*)*, which allow a prover P to demonstrate knowledge about a secret to a verifier V, without leaking anything about the secret itself [23, 35]. A ZKP must satisfy the following three properties [35]:

1. Completeness: if the statement is true, the honest V will accept it.
2. Soundness: if the statement is false, it is not possible to convince an honest V that the statement is true.
3. Zero-knowledge: no malicious V can learn anything about the secret from the proof, except for its correctness.

ZKPs provide probabilistic soundness guarantees, i.e., the probability that a malicious prover can craft a valid proof for a false statement that an honest verifier will accept is negligible but non-zero. In REV systems, ZKPs can be used to prove that a ballot is valid and well-formed without leaking anything about its contents and also to prove the correctness of decryption without revealing the private key used.

Interactive ZKP are formulated as a challenge-response game in which P first commits to a statement, and secondly, the V sends a challenge to the P, to which P finally responds. These three-move protocols are referred to as Σ-protocols [23, 35]. The communication overhead in these protocols is not suitable for many use-cases. The so-called *Non-Interactive* ZKP (NIZKP) do not require interaction and can be generated by the P alone, while the same goes for the verification by the V, which does not require the P.

The so-called Fiat-Shamir Heuristic (FS) is the most common technique to render interactive Σ-protocols *non-interactive* by replacing the random challenge generated by the V with a hash of the algorithm's transcript, which then serves as the random oracle, i.e., the source of uniform randomness. This allows NIZKP to be easily generated and published, allowing anyone to verify a statement without information leakage.

Different protocols have made use of the FS heuristic to reduce communication overhead in REV schemes. Bernhard et al. [8] identified that two distinct applications of such heuristics exist in the literature: a strong variant (sFS) and a weak one (wFS). While sFS hashes the commitments and the statement, wFS *only* hashes the commitments. This can lead to unsound proofs in the case of malicious provers: i.e., proofs that do not actually prove the statement that they are supposed to prove.

NIZKPs are a central tool widely used in REV protocols in order to reduce trust assumptions and provide verifiable proof without leaking any information about the

data at hand. For instance, in HE-based schemes, ballot validity NIZKPs are used to prove that a ballot is formed correctly and does indeed contain a valid ballot (for instance, a 0 or 1 vote). Also, NIZKPs are used to prove the correct decryption of shared tallies, for instance, when multiple entities are in possession of a private part of the election key, which can be implemented using Secret Sharing techniques.

Secure Multi-Party Computations (MPC) allows the removal of any TTP from a distributed computation, enabling participants to compute the function on their private inputs with a cryptographic protocol. One example of an MPC is built on the idea of Secret Sharing [10]. With Distributed Key Generation (DKG), trust can be distributed, and the risk of one party decrypting individual ballots is mitigated. DKG can be applied with threshold cryptosystems, where a threshold of n out of k total amount of trustees is defined, which then collaborate to generate a new public key (i.e., the election key). In order to decrypt a ciphertext, collaborative decryption must be performed by a minimum of n participating trustees, which decrypt ciphertexts with their shared private key share, where the cipher decrypts to plaintext again with the last share submitted. With this approach, no single centralized trustee protects the secrecy of the voting, but it is split among multiple trustees.

Commitment Schemes (CS) are another essential building block for many cryptographic protocols. A CS is used to commit to a value v without revealing it at the time of commitment, but possibly revealing it later, i.e., a CS is useful when P must be prohibited from changing a value they have committed to, but the value should also stay private at the time of commitment [78]. For instance, in proof transcripts for NIZKPs, it is often necessary that P commits to a certain statement and later reveals that commitment (e.g., the order of candidates) [44].

As an alternative to using HE-based voting protocols, Mix Nets (MN) build the foundation of many REV systems. There are two types of Mix Nets: Decryption Mix Nets (DMN), which were first proposed by [16], and Re-Encryption Mix Nets (RMN), first proposed by [65]. Decryption Mix Nets (DMN) decrypt votes that are encrypted in as many layers as there are nodes in the DMN. Therefore, each layer corresponds to a key owned by a DMN node. Each node removes one layer of the encryption and permutes the input, repeating the process until the last encryption layer is removed and the plaintext result is revealed. Re-encryption Mix Nets (RMN) use multiple rounds of re-encryptions (or re-randomizations) of the ciphertexts, while only one decryption step is sufficient for decryption to the plaintext result. In RMNs, each mix node permutes the input and re-encrypts or re-randomizes the inputs to make them look completely different from the input. The last mix node performs a decryption step to recover the plain texts.

REV schemes make use of a wide range of cryptographic tools to construct voting protocols. From a practical point of view, security mechanisms based on well-established cryptographic methods, such as TLS (Transport Layer Security)-encrypted communication channels, and key management approaches are also required to be assessed closely. However, these mechanisms are not specific to REV and are in use in other areas as well, and the expenditure ultimately depends on the deployment use-case and budget of the REV system at hand. One thing that many of the security mechanisms do have in common with the REV specifics is the

underlying computationally hard problems that are discussed in the next subsection, after discussing the ElGamal cryptosystem in detail.

3.1 ElGamal Public-Key Cryptosystem

The ElGamal public-key cryptosystem was formulated in 1985 [28]. ElGamal is often used in cryptographic voting schemes because of the multiplicative homomorphism, which enables the encoding of votes in ciphertexts (cf. Sect. 3.2). Further, ElGamal offers semantic security (cf. Sect. 3), because ciphertexts are not deterministic but probabilistic. The computational hardness assumptions for ElGamal are based on the Decisional Diffie-Hellman (DDH) problem (cf. Sect. 3.2). The DDH assumption is commonly considered to hold in the subgroup of quadratic residues modulo a safe prime: $p = 2q + 1$ [11].

3.1.1 ElGamal Key Generation, Encryption, and Decryption

ElGamal is defined within the cyclic group $\mathbb{G}$. The cyclic group, $\mathbb{G}_q$ with prime order q, is defined over a finite field $\mathbb{Z}_p$, defined over positive integers modulo p, where p is an odd prime. The cryptographic operations are performed in a subgroup $\mathbb{G}_q \subset \mathbb{Z}_p^*$, a multiplicative cyclic group $\mathbb{G}_q$ defined over the finite field $\mathbb{Z}_p$. Alternatively, the cyclic group $\mathbb{G}$ can also be defined over an elliptic curve that is itself defined over a finite field [28].

In ElGamal, the public elements are $\mathbb{G}$, p, q and two independent generators g and h, where $g, h \in \mathbb{G}_q \setminus 1$. The private key sk is a random value: $\mathsf{sk} \in_r \mathbb{Z}_q^*$, where $q = \frac{p-1}{2}$. The public key pk is defined as follows:

$$(\mathsf{sk}, \mathsf{pk}) = (\mathsf{sk}, g^{\mathsf{sk}} \mod p)$$

The encryption of a plaintext message m is performed by using a random value $r \in_r \mathbb{Z}_q^*$.

$$\begin{aligned} E(m, \mathsf{pk}) = c &= (a, b) \\ &= (g^r \mod p, \mathsf{pk}^r \cdot m \mod p) \end{aligned}$$

The decryption of a ciphertext $c = (a, b)$ is performed by computing the multiplicative inverse of a^{sk} multiplied with b.

$$\begin{aligned} D(c, \mathsf{sk}) = m &= (a^{\mathsf{sk}})^{-1} \cdot b \mod p \\ &= (g^{r^{\mathsf{sk}}})^{-1} \cdot \mathsf{pk}^r \cdot m \mod p \\ &= (g^{r^{\mathsf{sk}}})^{-1} \cdot g^{\mathsf{sk}^r} \cdot m \mod p \\ &= (g^{r \cdot \mathsf{sk}})^{-1} \cdot g^{r \cdot \mathsf{sk}} \cdot m \mod p \end{aligned}$$

3.2 Message Encoding in ElGamal

The correct message encoding is crucial for voting systems because votes (i.e., messages) are encoded and then encrypted. In detail, messages in the ElGamal cryptosystem must be encoded, i.e., mapped with a reversible function to a group element, such that the DDH assumption holds [21, 27]. Otherwise, the ElGamal cryptosystem is not secure against chosen plaintext attacks (i.e., does not achieve `IND-CPA`). Hence, two major variants of message encodings are possible [26].

The first variant to perform a message encoding had been introduced in [21]. The core premise is to encode any message $m \in \mathbb{Z}_q$ as g^m. The corresponding ciphertext is given by $E(m, \mathsf{pk}) = c = (a, b) = (g^r, \mathsf{pk}^r \cdot g^m)$. For instance, the Provotum 2.0 system uses this message encoding format [49]. The encoding enables additively homomorphic operations making the ElGamal cryptosystem *additive*, instead of *multiplicative* homomorphic. A downside of this encoding is that the plaintext, resulting from the decryption, must be decoded by computing the discrete logarithm of g^m (i.e., $g^? = g^m$), although computing discrete logarithms is hard. It is only possible for a restricted message space $m \in 0, 1$ where brute-forcing or using the `baby-step-giant-step` algorithm is possible [27]. In the listing below, two ElGamal ciphertexts $E(m_1)$ and $E(m_2)$ are homomorphically summed. To achieve that, the components are multiplied, i.e., its exponents are added up, resulting in the equivalent sum as if the plaintexts m_1 and m_2 had been added together.

$$E(m_1 + m_2, \mathsf{pk}) = E(m_1, \mathsf{pk}) \cdot E(m_2, \mathsf{pk}) = (g^{r_1+r_2}, \mathsf{pk}^{r_1+r_2} \cdot g^{m_1+m_2})$$

$$m_1 + m_2 = \text{decode}(D(E(m_1 + m_2, \mathsf{pk}), \mathsf{sk}))$$

The second encoding variation ensures all messages are restricted to quadratic residues in a safe prime group $\mathbb{G}$ of order q. Using this approach, it is required to check that the equality: $m^q \mod p \equiv 1$ holds *before* a message is encrypted. If the equality does not hold, the message must not be encrypted [27]. A downside of the second variant is that the equality check needs to be performed *before* encrypting, thus increasing the computational overhead due to the modulo exponentiation. However, in real-world practical systems, the increase is negligible for a single operation.

3.3 Hard Problems and Post-Quantum Safety

Computationally hard problems (e.g., factorization of prime numbers as defined in RSA) are the backbone of public-key cryptography. These problems are easy to formulate but extremely hard to solve. For computationally hard problems, no algorithm will run in a reasonable amount of time [5], and they are often also referred to as *intractable* problems. Most REV systems use cryptographic primitives, which are in turn based on hard problems. BCs and DLs are also based on

modern cryptography and, thus, are also based on such intractability assumptions. It is crucial to assess these problems against current and future adversaries, especially with the advent of quantum computing, which may break certain hard problems. Considering BCs and DLs, data is persisted on publicly readable storage, which means that future adversaries could store that data now and break the underlying schemes in the future (e.g., store election data now and break ballot secrecy in the future). Therefore, these assumptions directly impact the long-term privacy and verifiability properties because they are often assuming hard problems. Hence, it is important to define the assumptions and hard problems that are considered by these primitives and the direct implications on the security notions.

Hard Problems Most voting schemes are based on public-key encryption methods and are thus based on assumptions of computational complexity [44]. Since every major public-key cryptosystem deployed today relies on either factoring (using the RSA algorithm) or the Discrete Logarithm Problem (DLP), systems using these schemes inherit the underlying assumptions. One of the key hard problems that modern cryptography is built on is the *Factoring Problem*, which is defined as finding the prime numbers p and q, given a large number $N = p \times q$. The widely known and used RSA algorithms are based on the factoring problem. The security of the factoring problem is based on the fact that factorization of prime numbers is hard, i.e., it is hard to decompose N as a product of prime numbers.

The *Discrete Logarithm Problem (DLP)* is similar to the factoring problem. DLP describes a *discrete* problem, which means it deals with integers, as opposed to real numbers (which would be called continuous). Further, it deals with a *logarithm*, i.e., the inverse function of exponentiation. Also, it deals with mathematical groups of prime order modulo p, which are a set of elements (typically numbers) that are related to each other according to well-defined rules written as Z_p^*. The DLP is defined as finding the y for which $g^y = x$, given a base number g within some group Z_p^*, where p is a prime number, and given a group element x.

The last group of hard problems is the Diffie-Hellman problems, which are a special variation of the DLP. The *Computational Diffie-Hellman (CDH)* problem is a variation of the DLP. The CDH describes computing the shared secret g^{ab} given only the public values of g^a and g^b, and not any of the secret values a or b. If an eavesdropper was to obtain g^a and g^b, it should not be possible for her to determine the shared secret g^{ab}. Lastly, the *Decisional Diffie-Hellman (DDH)* problem is stronger than the CDH. The DDH assumes that the attacker is not able to compute g^{ab} partially, given the values of g^a and g^b, i.e., the attacker cannot learn *anything* about the shared secret. To ensure that, the value of g^{ab} needs to be indistinguishable from a random group element.

The importance of these problems may not be obvious at first glance, but most of the cryptographic methods used in everyday communications and the Internet are based on using these hard problems. Therefore, the problems are *assumed* to be hard to solve. These assumptions build the foundation of all major infrastructure used in modern computer networks and in many voting protocols and REV systems and BC and DL networks. The implications of that are clear:

REV systems, BCs, and DLs alike base their security on public-key cryptography and hash functions. These primitives are used to authenticate transactions (or ballots) with signatures and are also used in communication channels and consensus algorithms to identify authorities (e.g., PoA). Collision-resistant hash functions are used to link together blocks of transactions, effectively forming the chain or ledger. Advances in quantum computing directly threaten these building blocks, since it is estimated that in the next 20 years, a powerful Quantum Computer (QC) will be functional enough to break current strong public-key cryptosystems by breaking the underlying hardness assumptions [30]. The research community is tackling these issues with many post-quantum cryptography efforts and various novel approaches for post-quantum encryption and signatures. For example, the NIST is conducting standardization processes to establish solid post-quantum public-key cryptography mechanisms [62].

4 Overview of REV Approaches

Over the years, many REV approaches have been developed, evaluated critically, and refined. Thus, it is essential to describe these approaches, outlining their core ideas and adding necessary elements on selected cryptographic elements which may not have been covered in Sect. 3, providing the required background to follow the principle approaches outlined hereafter. Inspired by [9, 13], the following figures' organization serves as an essential reference and role model to introduce REV schemes and their specific challenges. The following approaches highlight the inherent problems of creating an REV system in practice. Privacy, verifiability, and trusted parties are given access to overview major features, characteristics, and concerns that are still partially present. While the PBB is an essential part of any of these approaches (and REV systems in general), the following subsections focus on the approaches, regardless of the practical deployment methods used (e.g., be it a centralized server, a BC, or a DL). For a dedicated discussion of practically decentralizing any of these approaches, Sect. 5 outlines, discusses, and evaluates the potential directions and deployments.

4.1 Encrypt and Sign Approach

A basic approach toward an EV scheme combines an encryption scheme with a digital signature scheme, as outlined in Fig. 4. The encryption scheme provides Ballot Privacy, while the signature scheme ensures the data integrity of the sender's transmitted message and authentication. During ongoing voting, the secrecy of the voter's ballots is ensured. However, since the voting service has the private election key (i.e., the counterpart of the public election key used to encrypt the ballots), the digital signatures are directly linking the encrypted ballots to the voters [13].

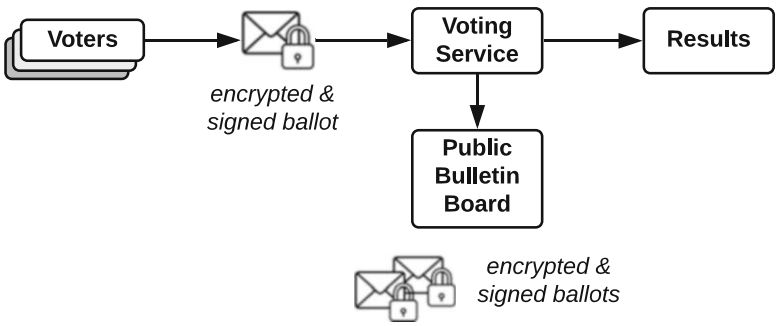

Fig. 4 Basic remote E-voting architecture based on a simple encrypt and sign

Thus, the voting service acts as a single trusted entity and can break ballot secrecy individually (e.g., decrypt individual ballots without anyone realizing). Also, a breach of the voting service's security could potentially leak the private key of the election and allow for complete decryption of the PBB, breaking ballot secrecy in full. By publishing a list of eligible public voter credentials, anyone can verify the signatures on the ballots. However, the major issue is that the final results cannot be verified without any additional mechanisms because the voting service would break ballot secrecy by decrypting ballots and publishing their plaintexts. While individual voters can verify whether their ballot was correctly stored on the PBB, they cannot verify the final results and fully trust in the voting service. In summary, the encrypt & sign approach is feasible for scenarios where voters choose to trust the voting service fully.

4.2 The Two Agencies Approach

Another popular voting scheme was first proposed in 1992 by [32], often referred to as the *two agencies model* [67], FOO92, or just FOO [9]. It is one of the first and most influential REV schemes designed for practical use. A simplified version of this voting scheme is shown in Fig. 5. The Validation Service authenticates the voter (checks eligibility) and then provides an anonymous token to the voter. The voter sends the encrypted ballot with the anonymous token to the Voting Service, which only appends the ballots if the Validation Service issued the anonymous token. The introduction of the validation service as a new TTP removes the link between the encrypted ballots and the voters' identities.

Voting schemes based on the two agencies model usually use the cryptographic primitive of Blind Signatures, which allows a signer to cryptographically sign a message without viewing its content in plaintext. Blind Signatures were first introduced in [16] as a means to establish untraceable and anonymous electronic payments. In the context of Fig. 5, the Validation Service can digitally sign the ballot

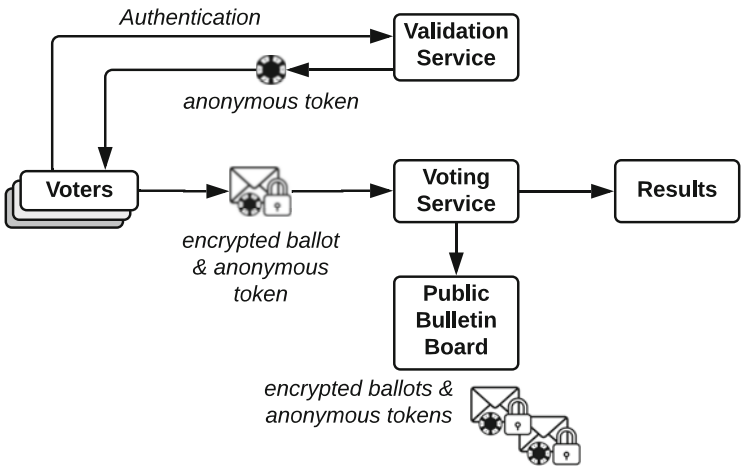

Fig. 5 Two agencies model with anonymous tokens (e.g., Blind Signatures)

without viewing its content (i.e., signing the blinded ballot). The voter receives the signed ballot, removes the blinding factor, and then casts the unblinded but signed ballot to the Voting Service, which validates whether the signature originates from the Validation Service. Even if the Validation Service colludes with the Voting Service, the token cannot be traced back to the voter since it was signed in its blinded form. Ballot Secrecy is assured in the final tally along the same line of arguments because the link between the vote and the token is removed. However, collusion between the two trusted agencies would make it rather easy to identify voters and break Ballot Privacy by correlating the information. Another major threat is *ballot stuffing* because the Validation Service could create valid tokens for non-eligible voters, which the Voting Service would accept. With this scheme, the voting service is still able to decrypt individual ballots since it owns the private election key.

4.3 Homomorphic Encryption Approach

Another major approach to achieve a private and verifiable REV is by deploying HE. In Fig. 6, an exemplary homomorphic tallying system is depicted. The ballot must be encrypted according to the format defined by the voting service. For instance, in this case, three ciphertexts of binary values, either 0 or 1, are encrypted three times, once for each voting option A, B, and C. Additionally, the ciphertexts may be cryptographically signed (for eligibility verification), and usually, the cryptographic proof is generated (e.g., a Non-Interactive Zero-Knowledge (NIZK) proof, more on that in Sect. 3). Because not a single ciphertext is decrypted during tallying (only the result), the proofs are crucial to ensure the ballot's validity and mitigate the risk that malicious voters could encrypt invalid values. Therefore, during tallying, the

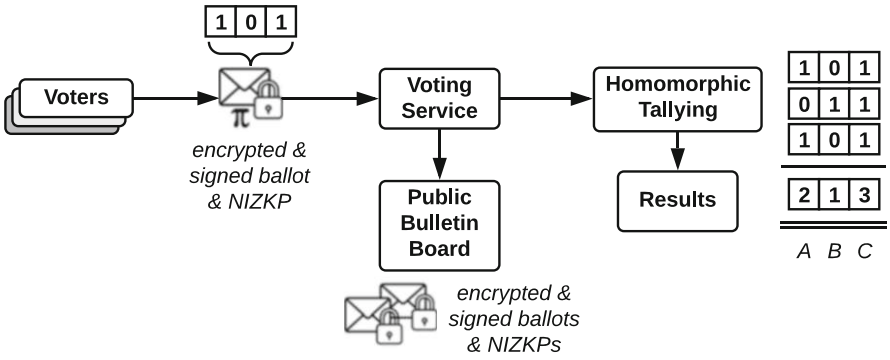

Fig. 6 Homomorphic Encryption-based REV system (e.g., ElGamal)

underlying cryptosystem's homomorphic properties are used to add the ciphertexts together and then only decrypt the final result. The major drawback of HE-based systems is that the number of possible voting options is limited (e.g., write-ins cannot be supported). A precise representation of the ballot is required.

Despite these limitations, an interesting advantage of HE-based systems is the straightforward distribution of decryption keys by using threshold cryptography. In such a case, each trustee holds shares of the private key, which is required to decrypt the final tally. In 1997, the seminal work [21] proposed the use of threshold cryptography in an REV system considering a multi-authority setting that requires each key share holder to submit partial decryptions. Then, the final decryption can only be performed if the pre-defined threshold of authorities collaborates.

An example voting system as depicted in Fig. 6 usually requires trustees to perform a Distributed Key Generation (DKG) algorithm, which results in a public election key pk_e, while each trustee holds a secret key share sk_e. Then, voters form a valid ballot and encrypt their ballot by using pk_e. Further, voters generate a NIZK proof proving that the ballot is well-formed and valid before sending both to the voting service. Upon receiving the ballots, the voting service authenticates voters and validates the proofs, ultimately storing them in the PBB. After the voting period ends, the homomorphic tallying is performed among all trustees over the set of all stored and valid ballots. For that, the trustees use their secret key shares to homomorphically tally the ballots together (without decrypting individual ballots), only decrypting the final tally and publishing it to the public.

4.4 Mix Network-Based Approach

Besides HE, Mix Networks (MN) are the primary building blocks for deployed REV systems used in real-world elections in Switzerland, Australia, Estonia, and Norway [40]. MNs are versatile cryptographic protocols that can be tailored to

other privacy-sensitive use-cases, such as anonymous communication protocols for messaging or routing.

In principle, MN-based approaches are similar to paper-ballot voting schemes where the paper ballots are shuffled after they are separated from any voter-identifying information. As seen in Fig. 7, once the ballot is no longer traceable through the MN, any correlation of the ballot before and after the MN is impossible. For the final tally, each ballot is decrypted individually in contrast to HE, where only the ciphertext of the final tally is decrypted. Further, MN-based schemes can distribute trust by leveraging threshold cryptography (similar to HE-based schemes cf. Sect. 4.3), for instance, by using multiple trustees to perform the MN re-encryption and final decryption ceremony.

4.5 *Discussion of Approaches*

The majority of current REV systems are based on either a MN- or a HE-based approach. In Table 2, an overview of various REV systems is provided, including academic implementations, enterprise and commercially oriented products, and country-specific implementations. The table is based on [6, 79], but focused only on REV schemes, with additional properties added, and it assesses the REV systems with regard to their properties based on the publication references listed alongside them. Most of these voting systems are evolving and being researched actively. Thus, novel improvements may be performed, and the table may not present a static view of these properties and systems.

The majority of REV systems provide privacy, although the details and assumptions vary depending on the voting protocol and implementation. Some systems also provide mechanisms enabling RF and CR. Often, these mechanisms require additional procedures or additional TTPs and services that perform specific actions, such as in [14] where one additional assumption is the existence of a re-randomization service that provides a re-encryption of ciphertexts so the voter does not possess a receipt of the encrypted ballot, but is still assured that the ballot is valid. Systems that provide CR, such as Civitas [19], allow voters to create fake credentials (which are not excluded in the final tallying) that a coercer cannot distinguish from real

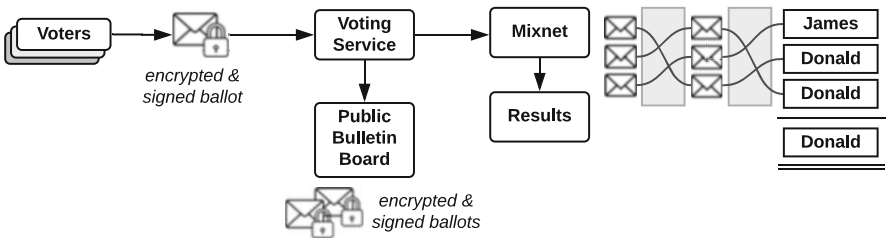

Fig. 7 Mix network-based REV system

Table 2 Comparison of related work on REV systems ● = Provides Property, ○ = Does Not Provide Property ◐ = Partially Provides Property. A Full Overview is Available in [8]

	Ballot Privacy	Receipt Freeness	Coercion-Resistance	Everlasting Privacy	Unconditional Privacy	Software Independence	Individual Verifiability	Universal Verifiability	Eligibility Verifiability	Cast-as-Intended	Recorded-as-Cast	Tallied-as-Recorded	Transparency	Decentralized PBB	Voting Protocol
Academic Implementations															
Helios [2]	◐[1]	○	○	○	○	●	●	●	○	●[2]	●	●	●	○	HE
BeleniosRF [14]	●	●	○	○	○	●	●	●	●[4]	○	●	●	●	○	HE
Civitas [19]	●	●	●	○	○	○	●	●	●[4]	○	●	●	●	○	RMN
Selene [71]	●	●	○	○	○	●	●	●	○	●[5]	●	●	◐[6]	○	RMN
Verify-Your-Vote [15]	●	●	●	○	○	●	●	●	●[4]	○	●	●	◐[6]	●	HE
OVN McCorry [57]	●	○	○	○	○	●	●	●	◐[7]	○	●	●	●	●	HE
Provotum 2.0 [48]	●	○	○	○	○	●	●	●	●	○	●	●	●	●	HE
Æternum [46]	●	◐[3]	○	●	●	●	●	●	●	○	●	●	●	●	CS
Enterprise Implementations															
nVotes [61]	●	○	○	○	○	●	●	●	○	●[2]	●	●	○[8]	○	RMN
Voatz [77]	○	○	○	○	○	○	○	○	○	○	○	○	○[8]	◐[12]	n/a
Country-specific Implementations															
CHVote [35]	●	○	○	○	○	○	●	●	●[4]	●[9]	●	●	●	○	RMN
Estonia [41]	●	○	○	○	○	○	●	●	○	●	●	○	◐[10]	○	RMN
Norway [32]	●	○	○	○	○	○	●	○	○	●	○	◐[11]	●	○	HE

[1] Vote copying possible
[2] Cast or challenge
[3] Extension available in [54]
[4] Uses trusted registrar
[5] Used e2e tracking number
[6] No source code available
[7] Needs trusted contract creator
[8] No source code, no documentation available
[9] Mailed verification code
[10] Partial source code available
[11] Needs trusted auditor
[12] BC deployed in private cloud

credentials. Besides coercion, long-term privacy, such as EP, is a major challenge for REV systems because most schemes are based on public-key cryptosystems, and privacy assurance relies on computational hardness assumptions (cf. Sect. 3.3). Also, achieving UP, i.e., not assuming any trusted authority for Ballot Privacy, is arguably impossible in practical HE or RMN systems since the decryption keys are split among trusted authorities. Also, protocols that provide UP rely on the existence of an anonymous channel to cast votes [48, 56] while still relying on intractability assumptions for verifiability properties (i.e., during the vote casting period).

Verifiability is a core property of REV systems, and its sub-properties are fundamental to such systems. As outlined in Sect. 2.3.2, if a system achieves IV and UV, that does not automatically imply that the system provides E2E-V. Achieving IV is done by enabling voters to check the correct inclusion of the ballot on the PBB. This creates tension with RF. UV is usually achieved by providing publicly verifiable proof that the decryption was performed correctly. Hence, ballots from the PBB were included and tallied correctly. The main challenge in achieving E2E-V is providing the ability for voters to check whether their vote was CaI.

One popular way to achieve this is the *Challenge-or-Cast* method, where a voting device's ballot encryption can be challenged by sending the ballot's parameters to a second trustworthy device that is used to re-perform the encryption steps. This verification method is used in [63]. Another option to achieve E2E-V includes *Code-Voting*, which requires a trustworthy printing office, as used in the CHVote voting system [37].

Another key aspect of REV systems is Eligibility Verifiability, which is tightly coupled to the IdM of voters, i.e., whether or not it is possible to verify whether eligible voters only vote once. Usually, REV systems assume external ballot authentication systems, i.e., a TTP that authenticates ballots by checking eligibility or distributing the credentials before the voting occurs. The variety in approaches also stems from IdM systems' heterogeneity across industries, use-cases, and geographical regions. Only a few Internet voting schemes have explicitly incorporated Eligibility Verifiability and provided defenses against ballot stuffing, e.g., JCJ [46] and Civitas [19], as well as Cobra [29].

The Transparency assessment in Table 2 mainly concerns the publication of the system's source code and additional documents that describe the voting protocol (e.g., security proofs). Transparency is crucial for trust in any REV system. In Switzerland, the Swiss Federal Government published an Ordinance [25] that demands an REV system publish its source code if it is used within a national context, with differing tiers of electorate percentages that would use said REV system.

Further, the deployment of decentralized PBBs is not widely adopted. Most of the REV systems included in Table 2 rely on centralized PBBs deployed as databases and, thus, a strong trust assumption in whoever is in charge of the database operation. Decentralized PBBs are, for instance, deployed on the public permissionless Ethereum mainnet [59] or deployed in a public permissioned PoA Ethereum network [15]. Also, dedicated DL and BC developments are proposed as decentralized PBBs [15, 49], which significantly improve the trust assumptions, i.e., instead of trusting one operator, trust is decentralized among multiple.

Voting protocols are the basis of the voting systems, and the majority of systems either rely on homomorphic or MN-based tallying. There are multiple key differences between HE- and MN-based approaches. First, with HE, the ballots' public aggregation is performed into the final outcome, whereas with an RMN, trustees (i.e., trusted mixers) anonymize the ballots by performing re-encryption. Also, with HE, only the outcome is decrypted, which means that little information is revealed and little computation is needed, whereas in MN-based tallying, more info is revealed, and the computation grows with the number of voters. Another key difference is the ballot validity checks. In HE-based schemes, the ballot validity can be checked at submission time because the voters perform heavy computations to generate the proof for ballot validity before casting. The proof needs to be adapted according to the election rules. In MN-based schemes, the validity check is performed *after* decryption, which means that the voters do not need to provide a ballot validity proof, and a universal ballot format can be used because ballots will be decrypted after mixing. In conclusion, the major burden in HE-based schemes

lies with the voters who need to prove the validity and ensure the ballot is cast correctly to the PBB. In MN-based schemes, the major burden is on the voting authorities who will perform the mixing. Another major responsibility is with the trustees that need to perform the mixing and decryption [12].

5 Research Directions Toward a Decentralized REV System

The primary goal of REV systems is to achieve a privacy-preserving and end-to-end verifiable voting process. Those approaches, as outlined in Sect. 4, show various directions to reach an overall secure process. While the primary responsibility of the PBB is to persist verifiable information immutably, the architectural model is not always considered carefully since the existence of such a PBB is either explicitly or implicitly stated as an assumption. Also, depending on the voting protocol chosen, the trust model is not influenced substantially either. For instance, if a single TTP stores private keys for a given election, the integrity depends solely on that TTP, whether the PBB is deployed in a distributed setting or not.

It is necessary to differentiate between a public permissionless BC or a public permissioned DL operating as a PBB since the underlying consensus algorithm's assumptions differ and impact key assumptions on trust (cf. Sect. 2.2). For example, a BC (e.g., Bitcoin) follows a permissionless, or generally spoken "trustless," and fully decentralized trust model, where it is not necessary to trust entities, only (i) their actions, which are publicly persisted on the BC and (ii) those cryptographic mechanisms in use. This implies a paradigm shift from placing trust directly in a centralized entity instead of trusting a decentralized platform's cryptographic mechanisms, which enables transparency and public verifiability. However, it is crucial to consider privacy aspects ensuring that Ballot Privacy (BP) is maintained under all circumstances, which means that, e.g., Bitcoin's fully public and permissionless approach becomes inadequate for guaranteeing scalable confidentiality.

Therefore, a generic centralized voting system's architecture is essential to take these considerations into account. Based on such a proposal as of below, its trust assumptions are analyzed, and a broader risk assessment is performed. This leads to a new, fully decentralized architecture proposal and a discussion of its trade-offs.

5.1 Centralized REV System

The architecture of a centralized REV system (cf. Fig. 8) is decoupled from a specific voting protocol. Major stakeholders include voters and one Voting Authority (VA). Voters are provided with the VA's voting credentials and are equipped with technical means to cast their ballots in their voting client. These credentials can be provided electronically (e.g., via electronic mail) or physically (e.g., via postal mail) by a VA for either mixing or decrypting the final tally of votes. In most cases,

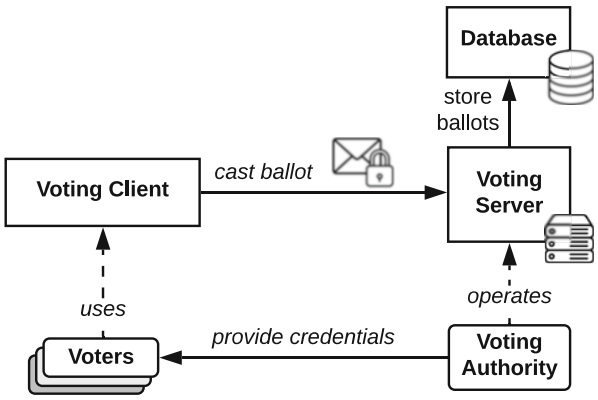

Fig. 8 Centralized REV system's architecture

a voting client is a software executed on voting clients, which can be considered personal devices owned by voters, e.g., typically Web browsers fetching the voting client software from a voting server. This allows for casting a remote vote in which the ballot is sent to the voting server to be persisted on a database that acts as a PBB.

For any properties to be evaluated, all trust assumptions need to be formulated beforehand. In essence, the centralized REV system requires a fully trusted VA that operates, stores, and manages cryptographic election keys to calculate the final tally. Further, the VA needs to operate a trusted server that persists all ballots cast and ensures the integrity and immutability of these ballots. While the server can be queried from the outside, the server's data is the single source of truth.

Assuming that a centralized REV system is based on a HE-based (Homomorphic Encryption) voting protocol, the homomorphic tallying process does not decrypt single ballots but homomorphically adds the tally and only decrypts the resulting tally. If the system is employing an MN (Mix Net) voting protocol, the mixing ensures BP. However, without any additional mechanism, the system does not provide Receipt Freeness (RF), because the voter can produce a receipt of the encryption procedure and its randomization parameters. Subsequently, Coercion-Resistance (CR) can also not be guaranteed since a coercer could force a voter to leak his/her credentials. Also, since the centralized VA generates the public election key, single ballots could be decrypted, thus, breaking BP altogether. While this threat can be mitigated by using threshold cryptography and splitting up the key to different trustees, the voting server's storage would still present a Single Point of Failure. For the decryption, the trustees would be required to interact with said voting server securely.

The ElGamal cryptosystem is the most popular choice for REV systems, used in both HE- and MN-based schemes. It bases its security on the DDH (Decisional Diffie-Hellman) problem (cf. Sect. 3.3). Therefore, the centralized approach does not provide Everlasting BP (EP), because an adversary can query the PBB, break the DDH problem in the future, and restore the private election key to decrypt all ballots to plaintext. Lastly, Unconditional Privacy (UP) is also not given since BP

relies on a trusted VA, i.e., if the VA is compromised or malicious, BP cannot be guaranteed. Thus, the *condition* for BP is trust in the VA.

All verifiability properties in any centralized approach rely heavily on trust assumptions, i.e., these properties *rely* on trusted VAs to persist and safeguard data integrity. For instance, to achieve Individual Verifiability (IV), voters can query the central database and check whether their vote is included in the set of ballots. Achieving Universal Verifiability (UV) is significantly more complex in a HE- or an MN-based scheme. For instance, the system can use a NIZKP to prove the correct decryption of the final tally or the final mix step, which the electorate can verify. Any centralized approach does not provide eligibility verifiability as such since it would require someone to verify that only eligible voters voted only once. Since the VA is trusted to provide credentials to all voters directly, the VA could create fake voters or cast votes with unused credentials. Without an Eligibility Verifiability, the final tally is not trustworthy. The first property of End-to-End Verifiability (E2E-V) is Cast-as-Intended (CaI) verifiability, which requires a method for voters to verify whether the vote had been cast as it was intended, i.e., whether the encryption was performed correctly by the voting device. The centralized approach does not provide any method to check whether the voting client's operations were correct and, hence, does not provide CaI verifiability at all. Nevertheless, the centralized approach can provide Recorded-as-Cast (RaC) verifiability since it can be checked by querying the central database and checking whether the ballot had been persisted correctly. Tallied-as-Recorded (TaR) (also referred to as Counted-as-Recorded, CaR) verifiability is given when the final proof of correct decryption verifies successfully.

5.2 Decentralized REV System

A major drawback of any centralized architecture is the dependency on trust on a single VA, i.e., the privacy and verifiability properties only hold under the assumption of a single trusted VA. Therefore, an approach for a fully decentralized and end-to-end verifiable architecture based on a public permissioned DL is provided here. The generic decentralized architecture for an REV system (cf. Fig. 8) leads to the observation that the significant difference of a centralized architecture is the decentralized deployment of the PBB, assumed to be operated as a public permissioned DL. On the voter's side, the process is similar to the centralized variant, but the voter *directly* interacts with the DL by running their own DL node or by communicating with a DL node operated by a TTP. Also, all VAs should operate DL nodes. These nodes form the public permissioned DL network, which the public can query, but only trusted VAs can sign and append new entries to.

The architecture (cf. Fig. 9) is agnostic toward any voting protocol selected. However, the voting protocol needs to include an approach that allows for the distribution of trust. By deploying a DKG scheme [33], the private election key is not held by a single authority. Finally, the authorities need to submit their decrypted

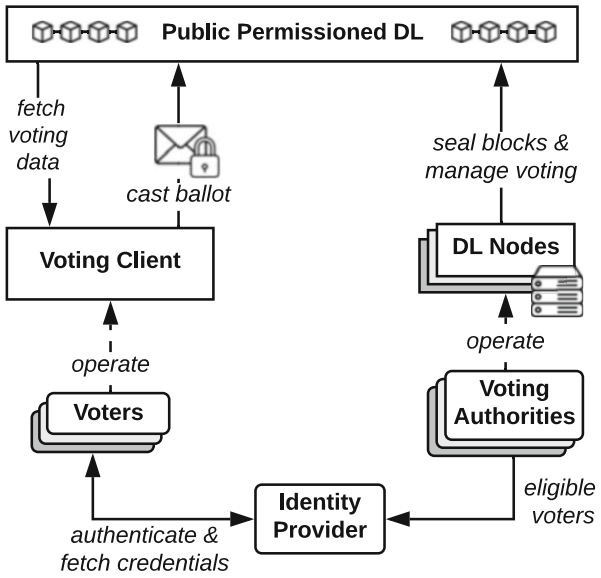

Fig. 9 A fully decentralized and generic high-level REV system's architecture

share of the final tally to decrypt the result collaboratively [50]. In that sense, the public permissioned DL serves as a transparent, tamper-proof platform, allowing the voting protocol to execute publicly shared information, which the electorate and auditors verify. Given a voting protocol based on a public key cryptosystem, voters encrypt their ballots with the public election key and send the encryption to the DL network directly, where anyone can verify the inclusion of the respective ballots [50].

The major benefit of using a decentralized architecture is the use of a suitable permissioned consensus algorithm to decentralize trust in the PBB. Many REV systems assume a centralized trusted PBB that provides a consistent view of the election's evidence trail. The implementation of cryptographic schemes into practical systems deployed on centralized PBBs does not hold under real-world conditions and breaks verifiability claims (e.g., integrity and availability guarantees). Thus, by decentralizing the PBB, the risk of a single entity breaking the integrity and consistency of the PBB is mitigated completely. The selection of an appropriate consensus algorithm is fundamental to the network's security characteristics [85]. Considering permissioned networks, where a set of initially trusted authorities is available (e.g., VAs, Identity Providers, and other TTPs), the Proof-of-Authority (PoA) consensus algorithm is suitable. Killer et al. [49, 50] proposes such a PoA-based PBB, allowing for the decentralization of trust. With a PoA consensus mechanism, the maximum number of tolerated malicious nodes is $\frac{N}{2}$ and $\frac{N}{3}$ for PBFT (Practical Byzantine Fault Tolerance), which is fully suitable for trusted VAs in operation. The selection of PoA is aligned to those REV system's general

requirements (cf. Sect. 2.4) since a set of trusted authorities is tasked with operating the voting or election. Thus, authorities allowed to propose and sign blocks (also called *minting*) are defined *a priori*, enabling a bootstrapping stage, where all authorities are already defined.

In summary, decentralizing the REV architecture allows for the distribution of trust on a public permissioned DL. The key benefit of a decentralized deployment is that trust assumptions are congruent with (i) trust assumptions of the voting protocol, i.e., threshold cryptography allows for distributing trust among VAs, which generates a public election key, as well as decrypting the final tally collaboratively. Secondly, benefit (ii) is that the trust is decentralized on a DL-consensus layer, since the PoA consensus only accepts blocks from pre-defined authorities. Thus, the decentralized architecture directly maps trust assumptions to a practically deployed system. Although these aspects present clear advantages, the translation of theoretical voting schemes to practical implementations always bears risks that require further research, e.g., there exist dedicated security threats on every layer of the system such that they need to be investigated in full detail before these systems reach fully practical feasibility.

5.3 Open Issues and Challenges

The previous section outlined a path toward a decentralized REV architecture. However, even with these major and measurable benefits of a decentralized PBB as identified, a couple of dedicated open issues remain:

- **Long-Term Privacy:** If the voting protocol's encryption scheme is based on a public-key cryptosystem (e.g., the ElGamal cryptosystem [28]), long-term privacy (e.g., EP and UP, cf. Sect. 2.3) issues need to be considered. For instance, the ElGamal cryptosystem bases its security on the DDH problem (cf. Sect. 3.3), which could be broken in the future [30]. Also, privacy is not provided unconditionally, because trusted authorities that hold a private key-share could conspire (or be compromised) and break BP.
- **Verifiability:** Achieving E2E-V in a decentralized architecture first requires a mechanism to achieve CaI verifiability (cf. Sect. 2.3), which allows voters to verify, whether the voting client correctly encrypted the ballot. The CaI verifier should be provided by DL nodes as well, decentralizing trust as well. Secondly, RaC verifiability can be checked by querying the PBB directly, without any TTPs. Similarly, TaR/CaR is achieved by storing the decryption proofs (i.e., including the necessary evidence trail) of the final tally on the public DL. Thus, anyone, including the voters, can verify that the steps were computed correctly. Attempts at manipulation would be immediately noticeable to any observer.
- **Voter Authentication:** Another crucial aspect of any REV system is ensuring that only eligible voters vote and that they only vote once. For instance, in Fig. 9, the Identity Provider (IdP) is responsible for verifying voter's eligibility. DL

nodes verify that each transaction containing a ballot originates from a public key through a blind signature provided by an IdP. Due to this mechanism, the IdP cannot trace an identity to a wallet address and a ballot. The IdP could also be decentralized in a way that is similar to the public permissioned DL by using threshold cryptography.

Further, real-world REV system deployments as partially discussed above made clear that open issues require future research to tackle interdisciplinary advances and challenges [8]. Future research requires the combination of theory and practice, combining both pillars of practical and theoretical research to formulate voting *protocols* in working *systems*. The *theory* pillar is formed by design techniques (e.g., Public-Key Cryptography [78]) and methodologies for verification (e.g., Applied pi calculus [72]) of voting protocols. The *practice* pillar combines implementation techniques (e.g., X.509 PKI [81]) with analysis methods (e.g., Attack Graphs [77]) toward practical systems [36] (Fig. 10).

On theoretical and practical pillars, open issues must be addressed before real-world REV becomes practically feasible. The following list of issues follows the structure of [8, 66], where many crucial and general security considerations for REV are stated. These aspects apply to all REV systems, whether votes are stored on either public permissionless BCs or permissioned DLs, since, either way, the PBB is only one aspect of the voting system as a whole, i.e., these security aspects are independent of the fact, whether the PBB is decentralized or not.

Generally speaking, all *stakeholders and adversaries* need to be clearly defined, including their capabilities, and responsibilities need to be assigned in full. The major formulation of trust assumptions builds the foundation for the system's security guarantees if any of the stakeholders is malicious. Hence, the fundamental questions include: *Who are the stakeholders and adversaries, and what are the trust assumptions, capabilities, and responsibilities within the voting process?*

Another high-level aspect is the *security model*, which needs to be carefully formulated in the context of the given voting system. A security model consists of security goals and a threat model, and these aspects are highly dependent on the relative importance of the election. Arguably, political elections should always

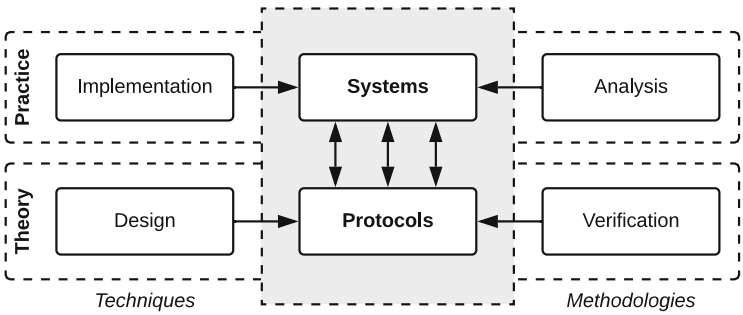

Fig. 10 Overview of research of REV, based on [36]

consider the mitigation of a state-sponsored and powerful attacker, while a risk assessment should always include attacks on all system areas. Hence, another crucial question is: *What are security goals and threat models the system needs to address, and under which trust assumptions do these goals hold?*

Furthermore, important questions regarding the key management deployment infrastructure need to be considered since they form the foundation of an REV system's operation that will be executed via the Internet. Additionally, key challenges in REV systems based on a fully decentralized PBB based on BC or DL must be considered carefully. First of all, (i) the decentralization of all single trusted entities should be enforced by the voting protocol and system. Second, (ii) inherent threats and risks of deploying a decentralized PBB (e.g., Distributed Denial-of-Service) requires specific mitigation techniques without introducing censorship. Third, (iii) the verification of the correct execution of the voting protocol and system should be provided on a decentralized PBB as a publicly verifiable audit trail. Possibly, a public and private environment is necessary to provide a suitable basis for [52]. Lastly, (iv) long-term privacy aspects (as well as vote-buying and coercion-resistance) are crucial and need to be addressed depending on the threat model and the election at hand because future adversaries may break computational intractability assumptions [48].

6 Overall Considerations

In an ideal election scenario, voters would express their voting choices remotely over the Internet and check each election stage transparently and independently. However, there are many challenges, ranging from societal to technological, which hinder the possibility of a remote election. Within the technical scope, the biggest challenge is to provide a solution that strikes a balance between privacy and verifiability embedded within the governmental and legal structure of a given democratic country. In this sense, this chapter explicitly outlined the current progress and the open challenges toward remote and decentralized elections.

Distributed Ledgers (DL), in their different flavors of permissioned and permissionless systems (including Blockchains – BC), stand out as a platform with the potential to provide fully decentralized elections, satisfying requirements in terms of privacy and verifiability based on existing cryptographic protocols. Public Permissioned DLs address a fundamental issue of REV systems' architectures: the strong assumption of a trusted PBB. REV systems require an immutable, publicly accessible, and verifiable audit trail. Hence, a permissioned DL can serve as a PBB for an REV system. A public permissioned DL's trust assumptions show a substantial improvement over a centralized PBB, where a single stakeholder needs to be trusted with the key management and operations. In contrast to a centralized deployment, a permissioned network can serve as a fully decentralized PBB, allowing for the distribution of trust on every layer of the voting system.

Transparency is a key aspect of public trust in the electoral process, and thus, a publicly readable DL can provide such an evidence trail.

> The peaceful transfer of power depends on confidence in the electoral process. That confidence should not automatically be given to any outcome that seems plausible – it must be earned by producing evidence that the election result is what the people chose. (Bernhard et al. [6])

However, there are major open issues in the interdisciplinary research of Electronic Voting in general. In particular, the constant threat of cyberattacks and security issues has created widespread awareness of cybersecurity issues. Even if all the crucial security considerations needed for high-stakes elections and voting systems are deployed, REV systems are still highly complex software/hardware systems that are practically impossible to deploy without any implementation errors. Therefore, it is fundamental that the REV system provides forms of dispute resolution and error detection. The software-independence property describes these idealized notions where implementation errors should be recognized effectively and do not go unnoticed. In essence, the cryptographic voting protocol should allow for implementation errors to be detected and optimally rolled back to recover the system's state. Also, more robust forms of verifiability, such as accountability, should incentivize stakeholders to act honestly. Hence, the critical challenges in REV remain the same: to strike a balance of privacy and verifiability while ensuring the integrity of the vote. Thus, future research on REV requires interdisciplinary approaches, combining theoretical voting protocols into practical and working systems, which decentralize the voting process, inherently reducing the potential abuse of power and trust. In an ideal election scenario, voters would express their voting choices remotely through the Internet and check each election stage transparently and independently. However, there are many challenges, ranging from societal to technological, which hinder the possibility of a remote election. Within the technical scope, the biggest challenge is to provide a solution that strikes a balance between privacy and verifiability embedded within the governmental and legal structure of a given democratic country. In this sense, this chapter aimed to explicitly outline the current progress and the open challenges toward remote and decentralized elections.

Acknowledgments This work was supported partially by (a) the University of Zürich UZH, Switzerland, and (b) the European Union's Horizon 2020 Research and Innovation Program under grant agreement No. 830927, the Concordia Project.

References

1. B. Adida, Advances in cryptographic voting systems. Ph.D. thesis, Massachusetts Institute of Technology, 2006
2. B. Adida, Helios: web-based open-audit voting, in *USENIX 17th Security Symposium (SS08)*, San Jose, CA, USA (2008), pp. 335–348

3. S.T. Ali, J. Murray, An overview of end-to-end verifiable voting systems. Computing Research Repository (CoRR) in arXiv (2016). https://arxiv.org/abs/1605.08554. Accessed on 11 Oct 2021
4. F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C.A. Reuter, M. Strand, A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192 (2015). https://eprint.iacr.org/2015/1192. Accessed on 11 Oct 2021
5. J.P. Aumasson, *Serious Cryptography: A Practical Introduction to Modern Encryption*. (No Starch Press, 2017)
6. J. Benaloh, M. Bernhard, J.A. Halderman, R.L. Rivest, P.Y.A. Ryan, P.B. Stark, V. Teague, P.L. Vora, D.S. Wallach, Public evidence from secret ballots. Computing Research Repository (CoRR) in arXiv (2017). https://arxiv.org/abs/1707.08619. Accessed on 11 Oct 2021
7. J. Benaloh, D. Tuinstra, Receipt-free secret-ballot elections, in *26th Annual ACM Symposium on Theory of Computing (STOC '94)*, Montreal, Quebec, Canada (1994), pp. 544–553
8. D. Bernhard, O. Pereira, B. Warinschi, How not to prove yourself: pitfalls of the Fiat-Shamir heuristic and applications to helios, in *Advances in Cryptology (ASIACRYPT 2012)*, LNCS, vol. 7658, eds. by X. Wang, K. Sako (Springer, Beijing, China, 2012), pp. 626–643.
9. D. Bernhard, B. Warinschi, Cryptographic voting – a gentle introduction, in *International School on Foundations of Security Analysis and Design VII (FOSAD 2012/2013 Tutorial Lectures)*, LNCS, vol. 8604 (Springer, 2014), pp. 167–211
10. G.R. Blakley, G. Kabatiansky, Shamir's threshold scheme, in *Encyclopedia of Cryptography and Security*, eds. by H.C.A. van Tilborg, S. Jajodia (Springer US, Boston, MA, 2011), pp. 1193–1194
11. D. Boneh, The decision Diffie-Hellman problem, in *3rd International Symposiun on Algorithmic Number Theory (ANTS-III)*, LNCS, vol. 1423, Portland, Oregon, USA (Springer, 1998), pp. 48–63. https://doi.org/10.1007/BFb0054851. Accessed on 11 Oct 2021
12. P. Bulens, D. Giry, O. Pereira, Running mixnet-based elections with helios, in *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE'11)*, San Francisco, CA, USA (2011)
13. S.G. Castelló, Individual verifiability in electronic voting. Ph.D. thesis, Universitat Politècnica de Catalunya, 2016. https://upcommons.upc.edu/handle/2117/96245. Accessed on 11 Oct 2021
14. P. Chaidos, V. Cortier, G. Fuchsbauer, D. Galindo, BeleniosRF: a non-interactive receipt-free electronic voting scheme, in *ACM Conference on Computer and Communications Security (CCS'16)*, Vienna, Austria (2016), pp. 1614–1625
15. M. Chaieb, S. Yousfi, P. Lafourcade, R. Robbana, Verify-your-vote: a verifiable blockchain-based online voting protocol, in *15th European Mediterranean and Middle Eastern Conference on Information Systems (EMCIS 2018)*, LNBIP, vol. 341, Limassol, Cyprus (Springer, 2018), pp. 1–14
16. D. Chaum, Blind signatures for untraceable payments, in *Advances in Cryptology (CRYPTO '82)*, eds. by D. Chaum, R.L. Rivest, A.T. Sherman, Santa Barbara, California, USA (1983), pp. 199–203
17. D.L. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2) (1981)
18. B. Chevallier-Mames, P.A. Fouque, D. Pointcheval, J. Stern, J. Traoré, On some incompatible properties of voting schemes, in *Towards Trustworthy Elections: New Directions in Electronic Voting*, LNCS, Vol. 6000, eds. by D. Chaum, M. Jakobsson, R.L. Rivest, P.Y.A. Ryan, J. Benaloh, M. Kutylowski, B. Adida (Springer, Berlin/Heidelberg, 2010), pp. 191–199
19. M.R. Clarkson, S. Chong, A.C. Myers, Civitas: toward a secure voting system, in *2008 IEEE Symposium on Security and Privacy (SP'08)*, Oakland, CA, USA (2008), pp. 354–368
20. J.D. Cohen, M.J. Fischer, A robust and verifiable cryptographically secure election scheme, in *26th Annual Symposium on Foundations of Computer Science (SFCS'85)* (IEEE Computer Society, Portland, OR, USA, 1985), pp. 372–382
21. R. Cramer, R. Gennaro, B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, in *17th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'97)*, LNCS, Vol. 1233, Konstanz, Germany (Springer, 1997), pp. 103–118

22. R.J. Dalton, The social transformation of trust in government. Int. Rev. Sociol. **15**(1), 133–154 (2005)
23. A. De Santis, S. Micali, G. Persiano, Non-interactive zero-knowledge proof systems, in *Advances in Cryptology: Proceedings of CRYPTO'87*, LNCS, vol. 293 ed. by C. Pomerance, Santa Barbara, CA, USA (Springer, 1988), pp. 52–72.
24. S. Delaune, S. Kremer, M. Ryan, Verifying privacy-type properties of electronic voting protocols: a taster, in *Towards Trustworthy Elections: New Directions in Electronic Voting*, eds. by D. Chaum, M. Jakobsson, R.L. Rivest, P.Y.A. Ryan, J. Benaloh, M. Kutylowski, B. Adida, LNCS, vol. 6000 (Springer, Berlin/Heidelberg, 2010), pp. 289–309
25. Die schweizerische Bundeskanzlei (BK): Verordnung der BK über die elektronische Stimmabgabe (VEleS) (vom 13. Dezember 2013 (Stand am 1. Juli 2018)) (2013). https://www.admin.ch/opc/de/classified-compilation/20132343/index.html. Accessed on 11 Oct 2021
26. M. Eck, Mixnets in a distributed ledger remote electronic voting system. Master's thesis, 2021
27. M. El Laz, B. Grégoire, T. Rezk, Security analysis of ElGamal implementations (2020), pp. 310–321. https://doi.org/10.5220/0009817103100321. Accessed on 11 Oct 2021
28. T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
29. A. Essex, J. Clark, U. Hengartner, Cobra: toward concurrent ballot authorization for internet voting, in *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/-WOTE 2012)*, Bellevue, WA, USA (2012), pp. 1–13
30. T.M. Fernandez-Carames, P. Fraga-Lamas, Towards post-quantum blockchain: a review on blockchain cryptography resistant to quantum computing attacks. IEEE Access **8**, 21091–21116 (2020)
31. C. Fontaine, F. Galand, A survey of homomorphic encryption for nonspecialists. EURASIP J. Inf. Secur. **013801**, 1–10 (2007)
32. A. Fujioka, T. Okamoto, K. Ohta, A practical secret voting scheme for large scale elections, in *Advances in Cryptology (AUSCRYPT'92)*, LNCS, vol. 718, Gold Goast, QLD, Australia (Springer, 1992), pp. 244–251
33. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007)
34. K. Gjøsteen, The Norwegian internet voting protocol, in *Third International Conference on E-Voting and Identity (VoteID '11)*, LNCS, vol. 7187, Talinn, Estonia (Springer, 2012), pp. 1–18
35. S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems, in *17th Annual ACM Symposium on Theory of Computing (STOC'85)*, Providence, Rhode Island, USA (1985), pp. 291–304
36. R. Haenni, E. Dubuis, U. Ultes-Nitsche, Research on e-voting technologies: a survey (2008)
37. R. Haenni, R.E. Koenig, P. Locher, E. Dubuis, CHVote protocol specification (Version 3.2). Cryptology ePrint Archive, Report 2017/325 (2017). https://eprint.iacr.org/2017/325. Accessed on 11 Oct 2021
38. T. Haines, M. Johannes, How not to VoteAgAin: pitfalls of scalable coercion-resistant e-voting. Cryptology ePrint Archive, Report 2020/1406 (2020). https://eprint.iacr.org/2020/1406. Accessed on 11 Oct 2021
39. T. Haines, S.J. Lewis, O. Pereira, V. Teague, How not to prove your election outcome, in *2020 IEEE Symposium on Security and Privacy (SP'20)*, San Francisco, CA, USA (2020), pp. 644–660
40. T. Haines, J. Muller, SoK: techniques for verifiable mix nets, in *33rd IEEE Computer Security Foundations Symposium (CSF'20)*. Virtual Conference (2020), pp. 49–64
41. J.A. Halderman, V. Teague, The new South Wales iVote system: security failures and verification flaws in a live online election, in *5th International Conference on Electronic Voting and Identity (VoteID'15)*, vol. 9269 (2015), pp. 35–53
42. F. Hao, P.Y.A. Ryan, *Real-World Electronic Voting: Design, Analysis and Deployment* (2016). CRC Press. https://www.amazon.com/Real-World-Electronic-Voting-Analysis-Deployment/dp/1498714692

43. S. Heiberg, T. Martens, P. Vinkel, J. Willemson, Improving the verifiability of the estonian internet voting scheme, in *International Joint Conference on Electronic Voting (E-Vote-ID 2016)*, LNCS, vol. 10141 eds. by R. Krimmer, M. Volkamer, J. Barrat, J. Benaloh, N. Goodman, P.Y.A. Ryan, V. Teague (Springer, 2017), pp. 92–107
44. H. Jonker, S. Mauw, J. Pang, Privacy and verifiability in voting systems: methods, developments and trends. Comput. Sci. Rev. **10**, 1–30 (2013)
45. H. Jonker, J. Pang, Bulletin boards in voting systems: modelling and measuring privacy, in *6th International Conference on Availability, Reliability and Security (ARES 2011)* (2011), pp. 294–300
46. A. Juels, D. Catalano, M. Jakobsson, Coercion-resistant electronic elections, in *ACM Workshop on Privacy in the Electronic Society (WPES'05)*, Alexandria, VA, USA (2005), pp. 61–70
47. A. Kiayias, M. Yung, Self-tallying elections and perfect ballot secrecy, in *International Workshop on Public Key Cryptography (PKC 2002)*, LNCS, vol. 2274 (Springer, 2002), pp. 141–158
48. C. Killer, M. Knecht, C. Müller, B. Rodrigues, E.J. Scheid, M. Franco, B. Stiller, Æternum: a decentralized voting system with unconditional privacy, in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC 2021)*. (IEEE, Darlinghurst, Australia, 2021), pp. 1–8
49. C. Killer, B. Rodrigues, R. Matile, E. Scheid, B. Stiller, Design and implementation of cast-as-intended verifiability for a blockchain-based voting system, in *35th Annual ACM Symposium on Applied Computing (SAC'20)*, Brno, Czech Republic (2020), pp. 286–293
50. C. Killer, B. Rodrigues, E.J. Scheid, M. Franco, M. Eck, N. Zaugg, A. Scheitlin, B. Stiller, Provotum: a blockchain-based and end-to-end verifiable remote electronic voting system, in *IEEE 45th Conference on Local Computer Networks (LCN)* (IEEE, Sydney, Australia, 2020)
51. C. Killer, B. Stiller, The Swiss postal voting process and its system and security analysis, in *International Joint Conference on Electronic Voting (E-Vote-ID 2019)*. LNCS, vol. 11759, Bregenz, Austria (Springer, 2019), pp. 134–149
52. C. Killer, L. Thorbecke, B. Rodrigues, E. Scheid, M. Franco, B. Stiller, Proverum: a hybrid public verifiability and decentralized identity management. Computing Research Repository (CoRR) in arXiv (2020). https://arxiv.org/abs/2008.09841. Accessed on 11 Oct 2021
53. S. Kremer, M. Ryan, B. Smyth, Election verifiability in electronic voting protocols, in *ESORICS. Lecture Notes in Computer Science*, vol. 6345 (Springer, 2010), pp. 389–404
54. R. Krimmer, A structure for new voting technologies: what they are, how they are used and why, in *The Art of Structuring* (2019), pp. 421–426. https://doi.org/10.1007/978-3-030-06234-7_39. Accessed on 11 Oct 2021
55. R. Küsters, J. Müller, Cryptographic security analysis of e-voting systems: achievements, misconceptions, and limitations, in *International Joint Conference on Electronic Voting (E-Vote-ID 2017)*, LNCS, vol. 10615 (Springer, 2017), pp. 21–41
56. P. Locher, Unconditional privacy in remote electronic voting theory and practice. Ph.D. thesis, University of Fribourg, 2016
57. P. Locher, R. Haenni, Verifiable internet elections with everlasting privacy and minimal trust, in *International Conference on E-Voting and Identity (Vote-ID 2015)*, LNCS, vol. 9269 (Springer, 2015), pp. 74–91
58. S.Z. Lowry, P.L. Vora, Desirable properties of voting systems. NIST E2E Workshop (2009). https://www.nist.gov/publications/desirable-properties-voting-systems. Accessed on 11 Oct 2021
59. P. McCorry, S.F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in *International Conference on Financial Cryptography and Data Security (FC 2017)*, LNCS, vol. 10322 (Springer, 2017), pp. 357–375
60. T. Moran, M. Naor, Receipt-free Universally-verifiable Voting with everlasting privacy, in *Advances in Cryptology (CRYPTO'06)*, LNCS, vol. 4117 (Springer, 2006), pp. 373–392
61. T. Moran, M. Naor, Split-ballot voting: everlasting privacy with distributed trust. ACM Trans. Privacy Secur. **13**(2), 246–255 (2007)
62. National Institute of Standards and Technology (NIST): Post-Quantum Cryptography. https://csrc.nist.gov/projects/post-quantum-cryptography. Accessed on 11 Oct 2021

63. nVotes: nVotes – Secure Online Voting Software. https://nvotes.com. Accessed on 11 Oct 2021
64. P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *Advances in Cryptology (EUROCRYPT'99)*, LNCS, vol. 1592, ed. by J. Stern, Prague, Czech Repbulic (Springer, 1999), pp. 223–238
65. C. Park, K. Itoh, K. Kurosawa, Efficient anonymous channel and all/nothing election scheme, in *Advances in Cryptology – (EUROCRYPT '93)*, LNCS, vol. 765, Lofthus, Norway (Springer, 1993), pp. 248–259
66. S. Park, M. Specter, N. Narula, R.L. Rivest, Going from bad to worse: from internet voting to blockchain voting. J. Cybersecur. **7**(1), (2021)
67. J. Puigalli, S. Guasch, Privacy and anonymity management in electronic voting. Identity Priv. Manag. **XI**(1), 59–95 (2010)
68. M. Richards, N. Ford, *Fundamentals of Software Architecture: An Engineering Approach* (O'Reilly Media, Inc., 2020)
69. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
70. R.L. Rivest, On the notion of 'software independence' in voting systems. Philos. Trans. R. Soc. Math. Phys. Eng. Sci. **366**(1881), 3759–3767 (2008)
71. P. Rogaway, The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162 (2015). https://eprint.iacr.org/2015/1162.pdf. Accessed on 11 Oct 2021
72. M.D. Ryan, B. Smyth, Applied Pi Calculus. Cryptol. Inf. Secur. Ser. **5**(2011), 112–142 (2011)
73. P.Y. Ryan, P.B. Rønne, V. Iovino, *Selene: Voting with Transparent Verifiability and Coercion-Mitigation*. LNCS, vol. 9604, Christ Church, Barbados (Springer, 2016), pp. 176–192
74. K. Sako, J. Kilian, Receipt-free mix-type voting scheme, in *Advances in Cryptology (EUROCRYPT'95)*. LNCS, vol. 921, Saint-Malo, France (Springer, 1995), pp. 393–403
75. F.B. Schneider, Open source in security: visiting the bizarre, in *IEEE Symposium on Security and Privacy (SP)*. Berkeley, CA, USA (2000), pp. 126–127
76. U. Serdült, M. Germann, F. Mendez, A. Portenier, C. Wellig, Fifteen years of internet voting in Switzerland: history, governance and use, in *2015 Second International Conference on eDemocracy & eGovernment (ICEDEG'15)*, Quito, Ecuador (2015), pp. 126–132
77. O.M. Sheyner, Scenario graphs and attack graphs. Ph.D. thesis, Carnegie Mellon University, 2004
78. N.P. Smart, *Cryptography Made Simple* (Springer Publishing Company, 2015)
79. R. Staubli, Secure deployment and configuration management for a decentralized remote electronic voting system. Master's thesis, 2021
80. C.J. Tolbert, K. Mossberger, The effects of e-government on trust and confidence in government. Publ. Adm. Rev. **66**(3), 354–369 (2006)
81. S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, Internet X. 509 Public Key Infrastructure (PKI) Proxy Certificate Profile. Technical report, RFC 3820 (Proposed Standard) (2004)
82. United Nations (UN) General Assembly: Universal Declaration of Human Rights (1948)
83. P. Vinkel, Internet voting in Estonia, in *Information Security Technology for Applications (NordSec 2011)* (2012), pp. 4–12
84. B. Witten, C. Landwehr, M. Caloyannides, Does open source improve system security? IEEE Softw. **18**(5), 57–61 (2001)
85. Y. Xiao, N. Zhang, W. Lou, Y.T. Hou, A survey of distributed consensus protocols for blockchain networks. Computing Research Repository (CoRR) in arXiv (2019). http://arxiv.org/abs/1904.04098. Accessed on 11 Oct 2021