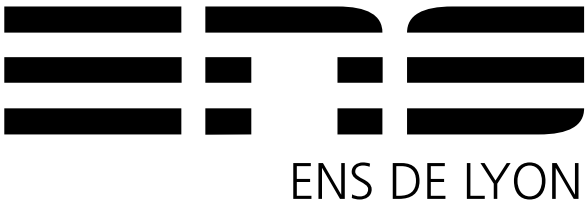
ROBERT RIEMANN

TOWARDS TRUSTWORTHY ONLINE VOTING: DISTRIBUTED AGGREGATION OF CONFIDENTIAL DATA

[14th November 2017 at 2:55 – version c082873]



THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON

OPÉRÉ PAR

L'ÉCOLE NORMALE SUPÉRIEURE DE LYON

DISSERTATION

Towards Trustworthy Online Voting: Distributed Aggregation of Confidential Data

PRESENTED BY
Robert Riemann

DIRECTED BY
Dr. Stéphane Grumbach

SUBMITTED TO
the doctoral school InfoMaths (ED 512) in partial fulfillment of the requirements for the degree of Doctor of Philosophy

DISCIPLINE
Computer Science

DATE
18 December 2017, Lyon

[14th November 2017 at 2:55 – version c082873]

Robert Riemann:
Towards Trustworthy Online Voting:
Distributed Aggregation of Confidential Data

[14th November 2017 at 2:55 – version c082873]

ABSTRACT

Aggregation of values that need to be kept confidential while guaranteeing the robustness of the process and the correctness of the result is necessary for an increasing number of applications. Various kinds of surveys, such as medical ones, opinion polls, referendums, elections, as well as new services of the *Internet of Things*, such as home automation, require the aggregation of confidential data. In general, the confidentiality is ensured on the basis of trusted third parties or promises of cryptography, whose capacities cannot be assessed without expert knowledge.

The ambition of this thesis is to reduce the need for trust in both authorities and technology and explore methods for large-scale data aggregations, that ensure a high degree of confidentiality and rely neither on trusted third parties nor solely on cryptography. Inspired by *BitTorrent* and *Bitcoin*, P2P protocols are considered.

The first contribution of this thesis is the extension of the distributed aggregation protocol *BitBallot* with the objective to cover aggregations in P2P networks comprising adversarial peers with fail-stop or Byzantine behaviour. The introduced changes allow eventually to maintain an accurate result in presence of an adversarial minority.

The encountered scalability limitations lead to the second contribution with the objective to support large-scale aggregations. Inspired by both BitBallot and BitTorrent, a novel distributed protocol called *ADVOKAT* is proposed.

In both protocols, peers are assigned to leaf nodes of a tree overlay network which determines the computation of intermediate aggregates and restricts the exchange of data. The partition of data and computation among a network of equipotent peers limits the potential for data breaches and reduces the need for trust in authorities. The protocols provide a middleware layer whose flexibility is demonstrated by voting and lottery applications.

RESUME

L'agrégation des valeurs qui doivent être gardées confidentielles tout en garantissant la robustesse du processus et l'exactitude du résultat est nécessaire pour un nombre croissant d'applications. Divers types d'enquêtes, telles que les examens médicaux, les référendums, les élections, ainsi que les nouveaux services de *Internet of Things*, tels que la domotique, nécessitent l'agrégation de données confidentielles. En général, la confidentialité est assurée sur la base de tiers de confiance ou des promesses de cryptographie, dont les capacités ne peuvent être évaluées sans expertise.

L'ambition de cette thèse est de réduire le besoin de confiance dans les autorités, de même que la technologie, et d'explorer les méthodes d'agrégations de données à grande échelle, qui garantissent un degré élevé de confidentialité et ne dépendent ni de tiers de confiance ni de cryptographie. Inspiré par *BitTorrent* et *Bitcoin*, les protocoles P2P sont considérés.

La première contribution de cette thèse est l'extension du protocole d'agrégation distribuée *BitBallot* dans le but de couvrir les agrégations dans les réseaux P2P comprenant des pairs adversaires avec un comportement défaillant ou byzantin. Les changements introduits permettent éventuellement de maintenir un résultat précis en présence d'une minorité adversaire.

Les limites d'évolutivité rencontrées conduisent à la deuxième contribution dans le but de soutenir les agrégations à grande échelle. Inspiré par BitBallot et BitTorrent, un nouveau protocole distribué appelé *ADVOKAT* est proposé.

Dans les deux protocoles, les pairs sont affectés aux noeuds feuilles d'un réseau de superposition d'arbres qui détermine le calcul des agrégats intermédiaires et restreint l'échange de données. La partition des données et du calcul entre un réseau de pairs équipotent limite le risque de violation de données et réduit le besoin de confiance dans les autorités. Les protocoles fournissent une couche middleware dont la flexibilité est démontrée par les applications de vote et de loterie.

PUBLICATIONS

The following publications contributed to the work presented in this dissertation:

Riemann, Robert and Stéphane Grumbach (2017a). 'Distributed Protocols at the Rescue for Trustworthy Online Voting'. In: *Proc. of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*. Porto, pp. 499–505. ISBN: 978-989-758-209-7. DOI: 10.5220/0006228504990505. URL: https://hal.inria.fr/hal-01501424.

— (2017b). 'Secure and trustable distributed aggregation based on Kademlia'. In: *IFIP Advances in Information and Communication Technology*. Ed. by F. Martinelli and S. De Capitani di Vimercati. Vol. 502. Rome: Springer. Chap. 12, pp. 171–185. ISBN: 978-3-319-58468-3. DOI: 10.1007/978-3-319-58469-0_12. URL: https://hal.inria.fr/hal-01529326.

— (2017c). 'Distributed Random Process for a large-scale Peer-to-Peer Lottery'. In: *Proc. of 17th IFIP Distributed Applications and Interoperable Systems*. DAIS'17. Neuchâtel: Springer, pp. 34–48. ISBN: 978-3-319-59664-8. DOI: 10.1007/978-3-319-59665-5_3. URL: https://hal.inria.fr/hal-01583824.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

CAS	central authentication service.
DDoS attack	distributed denial of service attack.
DHT	distributed hash table.
FPTP voting	first past the post voting.
KID	Kademlia leaf node ID.
PBB	public bulletin board.
PKI	public key infrastructure.
RPC	remote procedure call.
SMC	secure multi-party computation.

1

INTRODUCTION

In today's complex society, we often trust systems more than people.

— Bruce Schneier (2012)

A foundation pillar of our ever evolving society of increasing size and complexity is the cooperation among individuals. Cooperation relies on trust. In small communities, an important source of trust is the personal relationships among its members. However, with growing size, it becomes eventually infeasible to maintain personal relationships among all members. Cooperation in large communities may instead rely on *institutional trust*, i. e. trust in authorities. For instance, people may have trust in fair judgements of an independent law court without having a prior personal relationship to the judges. Similarly, people may have trust in the banking system, the police, social welfare system, and so on. Schneier (2012) argues that people feel easier nowadays to have trust in institutions than in people. Trusting authorities is reasonable if they are appointed by many people that enforce compliance using checks and balances. In this case, the risk of defects due to a dishonest minority is effectively lowered.

'Trust is a bet about the future contingent actions of others.'
—Piotr Sztompka

Where trust is insufficient or cannot be established, cooperation may be based on *security*, i. e. physical constraints on behaviour of people regardless their trustworthiness. In situations where the security mechanism is difficult to understand and to verify, trust in the capacity of the mechanism is required that shall be denoted *technological trust*. While online banking may in fact be secure, expert knowledge is necessary to verify the capacity of the algorithms that enforce security. With no expert knowledge, people rely again on trust in the technology or the assertions of trusted experts. Generally, more technology allows for larger cooperation and increase both the dependence on institutional and technological trust.

Cooperation can be characterised by its size measured in the number of involved members. Forms of large cooperation of global extension include international trade, intergovernmental organisations, international NGOs, as well as the Internet. The Internet again is the basis for social networks and, more generally, platforms, which govern cooperation and intermediation of unprecedented size. For instance, the social network Facebook (2017) reports of 2 billion monthly active members, which is approximately a quarter of the world population and presumably the majority of all internet users.

Large-scale cooperation entails more technological and often more institutional trust and thus bears an important risk. Dishonest author-

ities and technology failures may cause defects of global impact (Thielman and Johnston, 2016). This is in particular problematic where critical infrastructure is concerned, or individuals and their fundamental rights may be harmed. Nonetheless, large-scale cooperation emerges in the form of online services in all areas of life and is often greatly appreciated for its efficiency and convenience benefits. Prominent examples include the exchange of financial transactions *SWIFT*, the hospitality service *airbnb*, the online encyclopedia *Wikipedia*, and the marketplaces *Amazon* and *Alibaba*.

To reduce the risk of defects in online services, two distinct directions of ongoing efforts can be identified. First, new protocols of cooperation are conceived to increase the security and, if possible, rule out any risk by means of cryptography. For instance, encryption and digital signatures are used to ensure confidentiality, integrity and non-repudiation.

Second, the risk can be reduced by the separation of powers to multiple authorities or even to all members. As a result, the destructive potential of dishonest individuals is limited by their restricted power. This strategy is the basis for robust P2P protocols such as BitTorrent (Cohen, 2008) and Bitcoin (Nakamoto, 2008). Both require no authority and rely instead on the cooperation of equipotent members. For this, they assume trust in the set of all members. Some random dishonest members are tolerated and their destructive potential does not endanger the cooperation as a whole. Though, the majority of all members is assumed to be trustworthy.

The research subject of this thesis is online services that entail the accurate aggregation of confidential data. Its ambition is to explore methods for large-scale data aggregation, that ensure a high degree of confidentiality and rely neither on trusted third parties nor solely on cryptography. Hence, the need for trust in both authorities and technology is reduced.

The aggregation of confidential votes required to carry out votings is of particular interest. Various interest groups call for the introduction of voting methods, that meet the efficiency and convenience expectations established by many other online services (Chowdhury, 2014). Furthermore, voting presents an academic challenge (Bernhard et al., 2017). Depending on the use case, votings have very specific security requirement. Generally, voters shall not be compelled to rely on trust. Especially the losers of the vote must be provided evidence of the correctness of the outcome. Further, the vote of every voter shall remain confidential. Both requirements are in conflict with each other, because votes need to be shared in order to be counted. It seems that such a conflict cannot be resolved without trust (Chevallier-Mames et al., 2010). Thus, all solutions rely either on institutional or technological trust.

Ahead of its 2017 legislative elections, France drops online voting for citizens abroad over cybersecurity fears (Thomas and Stonestreet, 2017).

This contradiction between accurate and confidential aggregations is emblematic and occurs in an increasing number of applications. Various kinds of surveys, such as medical ones, opinion polls, referendums,

elections, as well as new services of the Internet of Things, such as home automation and smart metering, require the accurate aggregation of confidential data (Khan et al., 2012).

To understand the potential of cryptography and P2P networks for confidentiality-preserving applications, this thesis reviews existing solutions and explores novel approaches with a focus on voting.

Voting is the main confidentiality-preserving application realised with pre-digital technologies with its roots reaching back to the 7th century BC. It is an essential foundation pillar of every democracy and underlies generally strict regulations that impose the secret of the ballot and disallow the use of trusted third parties. The digitalisation of voting is subject of on-going scientific research since the 1960s and enjoys occasionally the attention of the general public, as most citizens of democracies are already familiar and concerned with voting procedures. Systems allowing for trustworthy electronic voting have long been and still are an open research question. Remote electronic voting, online voting for short, imposes even more challenges as its security must be entirely based on technology. In summary, the comprehensive material concerning the case of voting is a vivid source to study the trust assumptions of systems that rely on the aggregation of confidential data.

'It's not the voting that's democracy, it's the counting.'
—Tom Stoppard

1.1 PROBLEM STATEMENT

Contradicting requirements render the task difficult to aggregate confidential data as it is required for online voting. The aggregation of votes must be provably confidential and accurate without involving trusted authorities. While this has been achieved in the past with a paper-based scheme allowing for a verification of the ballot transport by eye-sight from the casting to the tallying, a different verification principle is required for online voting based on electronic ballots. The fast and efficient transport of electronic ballots and their automated aggregation allow to improve the convenience of voting, e. g. by remote ballot casting. At the same time, electronic ballots entail the risk of unauthorised overhearing and manipulation during their transport. A verification by eye-sight of electronic ballots is not possible any more.

So far, most propositions for secure online voting are based on advanced cryptography, which protects the secret of electronic ballots by their encryption (Bernhard et al., 2017). In consequence, *institutional trust* is necessary where authorities have the ability to decrypt individual ballots, and *technological trust* where the employed technology requires expert knowledge to assert its capacity. Authorities are further problematic as they constitute a single point of failure weakening the resilience of the protocol. Recent advances in cryptography allow to lower the impact of individual authorities at the expense of an increasing protocol complexity. The latter not only raises the barrier for

'The concept of absolute security does not exist. There will always be bugs.'
—Linus Torvalds

voters to verify independently the procedure, but also increases the chance for unintentional flaws in the implementation.

The sketched issue applies to many more services that generate personal data (e. g. Internet of Things) or require personal data (e. g. Artificial Intelligence). These services rely on the accuracy of their databases that must be assured despite the confidentiality constrains imposed inter alia by data protection regulations on the collection and manipulations of personal data.

1.2 THESIS STATEMENT

Distributed protocols such as BitTorrent, Bitcoin (cf. Section 3.2) and BitBallot (Chapter 4) demonstrate how trusted authorities can be omitted due to distributed computing while still maintaining an interesting set of security properties, that are ensured with high probability.

> *We claim that distributed protocols are promising to carry out trustworthy confidential aggregations.*

1.3 CONTRIBUTIONS

The contributions of this thesis are threefold.

1. In Chapter 3, the landscape of existing aggregation and voting protocols has been studied with a focus on distributed protocols. The development of distributed protocols for the purpose of online voting is motivated i. a. on the basis of their similarity to well-established paper-based voting methods (Riemann and Grumbach, 2017a).

2. Based on these findings, the patented protocol BitBallot (Reimert et al., 2016) and its design decisions are in Chapter 4 for the first time described for a broader academic audience. It allows to carry out aggregations of confidential data without cryptography, but is vulnerable to insider adversaries. Extensions are developed to improve the robustness and security of the protocol in case of adversaries. The scalability remains an open issue.

3. The experience with BitBallot led to the development of a novel distributed protocol ADVOKAT (Riemann and Grumbach, 2017b) described in Chapter 5. Routing based on the Kademlia DHT (Maymounkov et al., 2002) known from BitTorrent allows to improve its robustness and scalability. The Chapters 6 and 7 deal with two example applications of ADVOKAT, notably a voting application and a lottery (Riemann and Grumbach, 2017c).

To begin with, the following Chapter 2 gives a broader overview on voting requirements and technology including a historical summary.

2 BACKGROUND

A properly administered Australian paper ballot sets an extremely high standard that any competing election technology must match.

— Jones (2001)

Nearly all countries in the world carry out elections in one form or another. Consequently, a broad majority of the world population is familiarised with terms and concepts related to voting, either by school education, press coverage, or personal experience. Nevertheless, a disambiguation is provided to establish a common language throughout this work.

A voting is a method for a group to make a choice out of several options. An election is a particular voting to determine individuals to hold office. A *poll* is different from a voting in that not necessarily all group members have been inquired, but possibly only a non-representative subset. All three methods have in common the *aggregation* of individual choices.

The technical procedure employed to carry out a voting is called voting protocol. Most voting protocols consists of the following phases (Riera, 1998, Section 2; Aditya, 2005, Section 2.1):

REGISTRATION Individuals entitled to vote register as *voters* at the voting authority. The set of all voters is called electorate.

CASTING During a given timespan, voters can cast their filled ballot into an official ballot box.

AGGREGATION (TALLYING) All cast ballots are aggregated to the so-called *final tally*.

EVALUATION Based on the final tally, the voting outcome is determined with respect to the employed voting system, e. g. by majority.

VERIFICATION Voting protocols may allow to verify the compliance of the voting procedures with the actual voting protocol. The verification may be limited to certain aspects.

The voting protocol for a voting must be chosen with respect to practical and legal considerations. Especially for political elections such as general elections for the legislature, regulations apply that require particular protocol properties to promote basic democratic principles. The

principles have constitutional status in the case of e. g. Germany and France.

> *The deputies to the German House of Representatives are elected in general, direct, free, equal, and secret elections.*
>
> — *German Basic Law* (2014, Article 38)

> *Suffrage may be direct or indirect as provided for by the Constitution. It shall always be universal, equal and secret.*
>
> — *Constitution of France* (1958, Article I.3)

Moreover, the properties of political elections are affirmed by international declarations or conventions such as the *Universal Declaration of Human Rights* (1948, Article 21.3) approved by 48 member states of the United Nations (UN), the *European Convention on Human Rights* (1950, 1. Protocol, Article 3) ratified by all 47 member states of the Council of Europe, and the *Document of the Copenhagen Meeting* (1990, Article 7) of the Organization for Security and Co-operation in Europe (OSCE) representing 57 member states. The latter demands for elections that are universal, equal, fair, secret, free, transparent and accountable. The concept of universal elections consists of the right to vote of all adult citizens with only few exceptions. It describes the constitution of the electorate which is not covered in this work. The other properties depend on the voting protocol and define the objective of any voting protocol to be suited for political elections.

In the course of this chapter, the difficulty to develop such protocols is introduced in Section 2.1 and light is shed on the historical development of voting protocols in Section 2.2. An overview of common voting systems is given in Section 2.3 and the most common voting protocol, that is paper-based voting, is recapped thereafter in Section 2.4. Then, the promises of online voting and current experiences are presented in Section 2.5. The chapter concludes with prospects for applications aside from voting in Section 2.6.

2.1 CHALLENGES

The development of voting protocols that suffice high standards proves to be very difficult. Most protocols that are currently in use rely either on ideas with their roots in the democracy of Ancient Greece in the 5th century BC or implement some properties such as secrecy or transparency only unsatisfactorily. The difficulty arises out of the particular adversary model and the essential, but often contradictory protocol properties (Adida, 2006).

2.1.1 *Adversary Model*

Votings are supposed to be conducted in a hostile environment of mutual distrust. Neither voting officers, nor candidates and their respective political parties, nor the individual voters should be entrusted the administration of the voting, because all of them may have a strong interest to change the voting outcome to their favour. As a result, many voting protocols provide for a system of checks and balances to foster trust on the basis of supervision and cooperation.

2.1.2 *Protocol Properties*

The desired properties that shall allow for a secure voting in compliance with legal requirements are contradictory. In the following, we introduce common properties of voting protocols (Lambrinoudakis et al., 2003, Section 2.1; Haenni and Spycher, 2011, Section 2.1) and address the arising conflicts afterwards. The properties can be divided into two major categories.

2.1.2.1 *Security-Related Properties*

CORRECTNESS The voting outcome is derived from the final tally that reflects exactly the aggregated choice of all voters. It depends on the following sub-properties:

a) *Democracy* or *Eligibility*. Only registered voters can vote and can cast at most one ballot.

b) *Integrity*. Casted ballots cannot be altered, deleted or substituted.

c) *Accuracy*. The final tally includes all valid ballots and excludes invalid ballots.

PRIVACY No one can obtain more information about the ballots and the voters than what can be inferred from the final tally. It depends on the following sub-properties:

a) *Secrecy*. One cannot conclude from a cast ballot its voter and vice versa.

b) *Receipt-Freeness*. No one can gain information (*receipt*) to prove to someone else that he or she voted in a certain way.

c) *Coercion-Resistance*. No one can cooperate with a coercer to prove that he or she voted in a certain way.

d) *Fairness*. No one can gain intermediate tallies before the casting phase is terminated.

The property *confidentiality* signifies a weaker form of privacy in which certain information can be deduced only by a restricted subgroup. Gen-

erally, the confidentiality of the ballot is fundamental to allow voters to express their personal preference. Coercion-resistance and receipt-freeness address the scenarios of coercion, vote buying, peer pressure, etc. by ruling out any interactive, respectively, non-interactive proof of the vote in a certain way and diminishing as a result the incentive for such voter manipulations based on rewards or intimidations.

VERIFIABILITY The correctness (with its sub-properties) of the final tally and the derived voting outcome can be verified. One distinguishes between:

- *Universal Verifiability*. Anyone can verify the correctness of the tally.
- *Individual Verifiability*. A voter can only verify the inclusion of its own ballot in the tally. While some protocols require voters to reveal their ballot in order to proof their objection, other protocols do not (*open objection*).

ROBUSTNESS The correctness and confidentiality of the voting cannot be compromised by a reasonable sized coalition of corrupt or unavailable officers, voters or broken voting equipment.

2.1.2.2 *System-Wide Properties*

Voting protocols have to offer furthermore few so-called system-wide properties to be suited in practice.

The UN Convention on the Rights of Persons with Disabilities ratified by most countries affirms in Article 29 the 'right of persons with disabilities to vote by secret ballot [...] without intimidation'.

VOTER CONVENIENCE In order to allow for a broad participation, the registration to vote and the casting of the ballot must be as convenient as possible. No expert knowledge or special equipment shall be required.

a) *Mobility*. The casting period and location give voters high autonomy to decide when and where to vote.
b) *Accessibility*. The voting procedure is adapted to voters of diverse mother tongues or potential disabilities.

FLEXIBILITY The voting protocol supports various ballot question formats such as ranking, rating or approval of options.

For instance, in the 2014 general elections of India, the largest democracy, over 800 million people registered to vote.

EFFICIENCY The scalability with respect to the number of voters must be considered for votings with many voters or authorities to determine the voting outcome in a reasonable short timespan using a reasonable amount of resources.

2.1.2.3 *Conflicting Properties*

The security properties are essential for a fair voting with a correct voting outcome despite the adversary model of a potentially hostile environment of mutually distrusting parties with intentions to manipulate.

If possible, voting protocols shall ensure all of them. However, some of these properties are contradicting each other.

In order to verify that every ballot in the ballot box has indeed been cast by an eligible voter as required by the *eligibility* property, one would need to identify the person that cast the ballot in question which is contrary to the *secrecy* property.

Further, *individual verifiable* voting protocols must provide a proof to the voter that his or her vote has been cast as intended and counted as cast. *Universal verifiability* allows to verify all cast votes. Nonetheless, the proof must be presented in a form that cannot be used to convince others to be comply with the requirement of *receipt-freeness* and *coercion-resistance*.

It has been shown that indeed universal verifiability and unconditional privacy can be achieved only under impractical assumptions, e. g. no absent voters (Chevallier-Mames et al., 2010). Consequently, all practical voting protocols seek for a suitable balance of security properties with respect to the intended purpose.

2.1.3 *Public Perception*

'[…] if the largest banks in the world transfer billions of dollars every day electronically we can use the same technology to ensure secure voting.'
—Phil Noble

The challenge to provide secure voting protocols, especially for the internet, is typically underestimated by the interested public (Schneier, 2001). Voting is compared to other security-sensitive activities, e. g. the online transaction of billions of dollars on a daily basis. Schneier points out two important differences, namely *anonymity* and *recovery*. Financial transactions are not anonymous. Sender and receiver are attached to each financial transaction, so that identifying fraud becomes easier. Moreover, chances are high that erroneous online financial transactions can be undone, funds be recovered, and criminal suspects may be brought to trial by the injured party.

In contrast, the privacy property of voting protocols prevents to track the individual casting of ballots, so that erroneous voting outcomes are more complicated to identify. Consequently, there is no obvious injured party and criminal suspect. Nonetheless, the voting protocol shall offer means to proof fraud and means to turn down wrongful claims of fraud. At last, redoing a voting is highly problematic in terms of fairness, because it is impossible to reproduce the conditions at the time of the original voting.

2.2 HISTORICAL CONTEXT

In the course of this section, a light is shed on the development of voting protocols in the face of growing size of communities and technological achievements that led to paper-based voting employed nowadays in many countries and presented hereafter in Section 2.4.

Figure 2.1: Voting with *psephoi* (pebbles) in a scene from the Wine Cup with the Suicide of Ajax. Red-figured *kylix* made in Athens about 490 B.C., attributed to the Brygos Painter. The J. Paul Getty Museum, 86.AE.286.

The subject of voting and its organisation is as old as democracy itself that can be traced back at least to Ancient Greece of the 7th century B.C. In this chapter, an overview of nine centuries of voting is presented from a technological point of view.

2.2.1 *Voting in Ancient Greece*

Voting procedures in Ancient Greece have been either directly described in works such as the *Constitution of the Athenians* of Aristotle (384 BC – 322 BC) or can be inferred from vase paintings and theatre plays.

The city-state Sparta in Ancient Greece in the 7th century BC was constituted as an oligarchy, which provided for the Council of Elders (*Gerousia*) that hold judicial powers and prepared legislation for approval by the Assembly (Girard, 2010). The members of the Council were elected by *acclamation voting* in which whoever was judged to have received the loudest acclaim was elected. To provide a fair, unbiased judgement of the loudness, few judges were appointed randomly from all voters and locked up in a room close to the assembly. Candidates were presented to the assembly one after another without speaking a word to prevent their identification by the judges. The favour of the assembly towards one candidate was then assessed by the judges who established a ranking of all candidates by the loudness of the assembly. This form of acclamation voting provides no privacy and only to a certain extend correctness, because the weight of the voters depend on their individual voice strengths. The judges had to be trusted to not know the order of presentation of the candidates.

Aristotle is credited to find the Spartan voting procedure 'childish'. In fact, he reports of voting procedures in Athens that are more elabor-

ated (Boegehold, 1963; Hansen, 1977). The procedures are supposedly influenced by the common practice of using pebbles to reckon complex sums when accuracy was wanted. The use of pebbles provides a visual representation of the sum that can be verified by recounts. In the vase painting (Figure 2.1), a voting scene is depicted. Under public supervision, men deposit pebbles on one of two distinct piles to express their support for one option. The piles are eventually counted and the option that received the most support in form of pebbles is chosen. As long as the number of voters is limited, so that double voting can be prevented, this protocol ensures correctness and is universally verifiable. However, the protocol is impractical in large elections. It is reported that also jars were employed to gather the pebbels for each option. The deposit is still easy to observe, so that no secrecy is provided. The hidden deposit, e. g. behind a curtain, would give rise to the danger of the deposit of more than one pebble per voter or the transfer of pebbles deposited earlier.

Lysias (445 BC – 380 BC) recognised the danger of reprisals that can follow votings lacking of secrecy and criticised his contemporary Agoratos for coercing the voters by enforcing an open vote. In 458 BC, the idea of secret voting was known in Athens. The jars were covered under a sort of wicker funnel with its large end down. After the voter was found to hold in its hand only one pebble, he or she reached into the funnel to place the pebbles in one of the jars. Bystanders may have been able to reveal the chosen jar by the clink of the falling pebble, so that the provided secrecy was imperfect.

A more advanced procedure was employed in the 5th century by the law court of Athens. An uneven number of up to 501 jurors would vote in favour of one of the two litigants. Therefore, all jurors were provided two nearly identical ballots of bronze that look like disks with short axes running through their centre. The both ballots differ only by the nature of their axe: one is hollow, the other solid. Taking in each hand one ballot with the axe ends covered by the fingers, the juror would cast one ballot into a bronze urn and discard the other one in a wooden urn. After all jurors cast their ballots, the ballots in the bronze urn were counted and the court decision found by majority. The procedure ensures correctness and to a great extend privacy. While jurors may have been incentivised to vote in a certain way, the enforcement requires a lot of afford, so that with limited resources the influence is limited as well.

Next to the law court, also the principle assembly of Athens (*Ecclesia*) carried out votings (Hansen, 1977). It is assumed that the assembly were usually attended by 6000 citizens that are supposed to have voted about 25 times during one session occupying not more than one day. With these limited time resources, the assembly opted to vote by the show of hands. Votings were conducted by stages. In case of a binary question, first the *ayes* and then the *nays* were asked to raise their hands.

Figure 2.2: The County Election (detail). Illustration of a polling place in Saline County, Missouri, US, in 1846 by George Caleb Bingham. Reynolda House Museum of American Article 1983.2.37.

The assembly sat divided in groups which allowed ten appointed officials to assess in parallel the majority, possibly most often by an estimate and only on request by a count. Hence, secrecy and correctness were presumably abandoned for the sake of a speedier voting in the face of the large number of voting matters and voters.

2.2.2 *Towards Secret Paper-based Voting*

With the transition to the medieval period and the dissolution of democracies, little progress in terms of voting technology have been observed until the early 19th century. The following presentation is based on the comprehensive website of Jones (2001, cf. 'A Brief Illustrated History of Voting').

Figure 2.2 depicts a voting in 1846 in the US. The man in the centre is swearing with the hand on the bible in front of the judge that he is eligible to vote and has not done so already. Afterwards, voters would announce their choice to the voting officers sitting behind the judge to record all votes next to the voter's names in their pollbook. Multiple pollbooks can be compared in case of objections. The procedure is similar to those based on pebbles from the 5th century depicted in Figure 2.1. Privacy is not ensured. Verifiability is given by public observation or consultation of the pollbooks.

The first use of paper-based voting happened presumably in Rome in 139 BC. The first use on American soil was presumable in 1629 to elect a pastor. Here, voters provided their own ballots which makes privacy difficult to achieve. In the beginning of the 19th century, parties started to encourage their supporters to use preprinted, partisan paper ballots and by the century, it become common to print on distinct striking paper in order to allow an easy observation of the cast ballots. The transparent ballot boxes, a measure against the prevailing issue of bal-

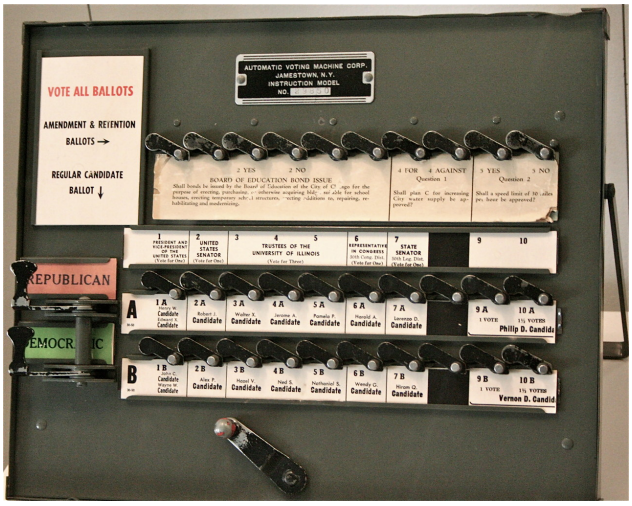


Figure 2.3: Lever voting machine of the Automatic Voting Corporation. Voters operate levers to answer several binary questions. Smithsonian National Museum of American History, Wikimedia.

lot box stuffing, rendered the task to deposit such paper ballot secretly virtually impossible.

A notable milestone paving the path towards secret paper-based voting is the pamphlet *People's Charter* published by the London Working Men's Association in 1838. Next to demands for voting rights, the importance of privacy was emphasised and an illustrated description of a voting procedure was provided. Voters would step behind a viewing barrier to drop a small ball, the *ballot*, into the one hole of the table that corresponds to their choice. The ball would then increase the counter of a mechanical counting mechanism and fall out of the table which is publicly observable and provides universal verifiability. The correctness relies on the well-functioning of the counting mechanism.

Ballot: from Italian ballotta *or Middle French* ballotte *for small ball.*

With increasing demand for secret votings, solutions were sought to achieve a secret voting without the need of such a special table. It was in Australia in 1858, that a secret paper-based voting was carried out using standardised paper ballots printed at government expense to be cast in simple ballot boxes. An in-depth discussion of paper-based voting is given in Section 2.4.

2.2.3 *Mechanisation*

While a properly administered paper-based voting offers a high standard in correctness, privacy and verifiability, there is a plethora of loopholes if that is not the case. Issues in the interpretation of hand-written marks on the paper ballots and erroneous tallying lead to the development of mechanical lever voting machines by the end of the 19th century. An example is depicted in Figure 2.3. By that time, lever voting machines were considered as high-tech with more moving parts than

almost anything else being made (Jones, 2001). By the mid 20th century, those machines were omnipresent in US elections and are still today in extremely widespread use even though the machine production stopped in 1982.

Lever voting machines eliminate all questions of ballot interpretation. By the end of the casting phase, the tallied result is directly given by the machine counters reducing greatly coincidental errors. A viewing barrier ensures privacy during the machine operation. The disadvantage of lever machines is its complexity that does not allow voters with no special knowledge to verify the proper functioning of the machine. Systematic counting errors of either benign or malign nature are difficult to assess. For instance, statistics show the counters are more likely to block at 99 than e. g. 98 or 100 due to mechanical limitations (Saltman, 2006). Further, the machines are potentially subject to malicious manipulation of the counting mechanism that may impact not only individual ballots, but all ballots cast using the respective machine. Once that doubts were raised, they are difficult to dispel, because ballots are not preserved individually, but instead directly tallied. A verification by recount is impossible.

To improve the verifiability and allow for a recount, procedures where thought of that conserve the individual ballot. In the 1960s, pre-scored cards were used as ballots. On their individual ballot card, voters punch out with a stylus behind a visual barrier one or more pre-scored holes corresponding to their choice. The punched ballot cards give less room for speculation of the voter intend and can be tallied automatically using pneumatic counting machines. In case of doubts, a manual recount of all ballots is possible. Nevertheless, reports were published by the mid 1980s calling to abandon voting with punched cards due to the high potential for systematically erroneous tallying in cause of accumulated paper chads blocking the counting mechanism (Saltman, 2006).

2.2.4 *Electronic Voting*

Another technique for automatic tallying is based on optical scanning of ballot cards that are either marked using a particular pencil (1930s) and later also ordinary graphite pencils (1960s). The tallying is carried out electronically. As such, *optical scan voting* can be considered as a form of electronic voting. While such systems are resistant to paper chad issues, the configuration of the electronic tallying may be prone to configuration mistakes and manipulations.

Direct-Recording Electronic (DRE) voting provides for a casting of the voter intent without physical support of a ballot such as pebbles, balls or paper. Instead, the voter intent is directly recorded by electronic means. The first proposal patented by Albert Henderson dates back to 1850 and comprises electrical telegraphy for casting. However, it was not before 1975, that a DRE voting machine called *Video Voter* was

used for the first time in a general election. DRE voting offers several advantages. The presentation of the (virtual) ballot can be adapted to the voters individual needs (e. g. their mother tongue, disabilities, etc.), assist voters during the casting to prevent invalid or incomplete votes and may provide after the casting an immediate feedback that their choice has been cast as intended. Further, there is no risk of exhausting the supply of physical ballots. DRE voting has the potential to ensure correctness and privacy. The downside of electronic voting is the verifiability. Most often, voters are required to trust the assertions of experts, e. g. of the voting machine developers or at best third-party auditors chosen by voting officers. The arising issues for the legitimacy of the voting outcome and potential technical solutions are at the core of this work. Regardless of the weak verification means of DRE voting, it has been adopted by large countries such as Brazil, India and the US due to its efficiency.

Some DRE voting protocols provide for each cast ballot a *voter verified audit trail* (VVAT), e. g. on paper. The trail allows voters to verify that their vote has been recorded as cast. In case of objects, the audit trail enables an entire or partial manual recount.

Voting protocols that provide for a transmission of the electronic ballot from one or more polling places to another tallying location over public network, e. g. the internet, are called *public network DRE voting* or online voting. Estonia became in 2005 the first country to run an online voting pilot project to vote for legally binding general elections.

While online voting allows for great voter convenience, it introduces new challenges to provide correctness and privacy that are addressed in detail in Section 2.5.

2.3 VOTING SYSTEMS

A voting system is a set of rules for translating sets of preferences as expressed in a voting into the actual voting outcome. To conduct a voting, a voting system must be chosen and further a suitable voting protocol that describes the procedures of voter registration, ballot casting and tallying, and their verification. Not every protocol is flexible enough to support efficiently all kinds of voting systems. If in Ancient Athens the voter preference was expressed by the deposit of one pebble in one out of many vases (cf. Section 2.2.1), one can easily determine the preference of the majority that corresponds to the vase containing the most pebbles. However, other voting systems may require voters to provide their preference by a ranking, i. e. an ordered list of options. Here, the deposit of one pebble in one vase is hardly sufficient to express the preference in a practical manner. In consequence, the choice for a suitable voting protocol does not solely depend on the protocol's properties, but also on the voting system that is employed.

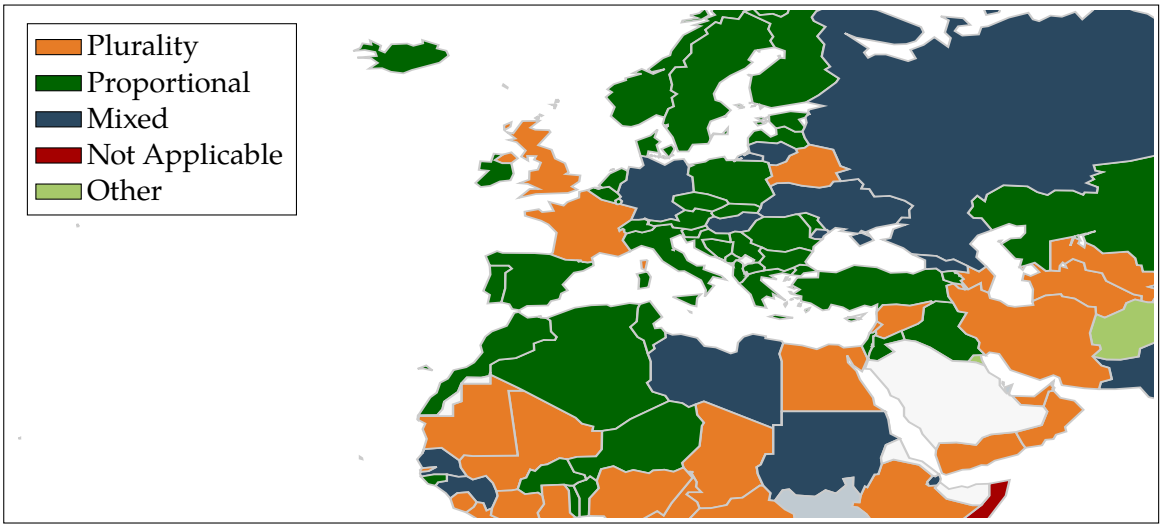


Figure 2.4: Voting systems for national legislatures. Voting systems of all three voting system families are employed in Asia, Europe and Africa. Source: http://www.idea.int/data-tools/question-view/130357 (visited on 15/04/2017).

Voting systems are conceived to strengthen different aspects, e. g. the presentation of minorities, voting outcomes backed by solid majorities, promotion of coalitions. Different voting systems may produce different voting outcomes even for the same sets of preferences, which is why the choice is crucial and affects the form of the voting questions to be answered. The form of the voting questions may constrain the number of suitable voting protocols.

A brief overview of voting systems commonly used for elections (cf. Figure 2.4) and their requirements is given hereafter based on the comprehensive handbook of electoral system design (Reynolds, Reilly and Ellis, 2005, Chapter 2). Without loss of generality, a voting is assumed to assign a subset from all n candidates, respectively parties, to a number of seats, e. g. in a parliament. Most voting systems are classified into three broad families based on two characteristics:

a) proportionality of received votes to accorded seats

b) level of *wasted votes* that do not help to elect a candidate/party

2.3.1 *Plurality Systems*

In plurality voting, also called majority voting, each voter expresses his preference by ranking, approving or selecting candidates that are elected with respect to received consent. Single-winner and multi-winner votings are supported. Candidates that have not received a sufficiently large number of votes are discarded. As such, the votes they received do not contribute to elect a candidate and are wasted.

Plurality voting systems belong to the simplest solutions and are widely deployed. For instance, 43 out of 193 UN member states use such systems for their national elections (*Comparative Data* 2014).

2.3.1.1 *First Past the Post (FPTP) Voting*

The name origins from horse races, where the one and only winner is determined by whom reaches at first the goal (the post). With respect to voting, the candidate is elected that received the highest number of votes, which does not necessarily represent the absolute majority. One speaks of *winner-takes-all*. The number of waste votes can be tremendous. For instance, if one candidate receives 20 % and each other candidate less, 80 % of the voters find their votes wasted. Voters have to chose one out of n candidates, which is supported by most voting protocols.

2.3.1.2 *Two-Round System (TR), Runoff Voting (RV)*

To elect a candidate with an absolute majority of at least 50 % of all votes, the *first past the post protocol* is carried out in two passes. If no candidate receives the absolute majority of all votes, a second pass takes place in which only the two most successful candidates stand for election. Here, the number of wasted votes is limited to 50 % in the worst case. For instance, France and Turkmenistan employ this system for their legislative elections.

2.3.1.3 *Alternative Vote (AV), Instant-Runoff Voting (IRV), Ranked Choice, or Preferential Voting*

The *two-round system* requires two separate passes if no candidate receives the absolute majority in the first pass. The voting lasts longer and is more expensive. To eliminate the need for the second pass, voters select not only one candidate, but instead rank all candidates with respect to their personal priority. Ballots are first evaluated according to the first preference. If there is no absolute majority, the candidates with the fewest votes are one by one excluded and their votes are redistributed according to their voters' second preference and decreasing until an absolute majority is found. The voting protocol must support ballots expressing an ordered list. E. g., the president of Ireland and the president of India are elected using the AV system.

2.3.1.4 *Cardinal Voting (CV)*

Cardinal Voting or *ranged voting* allows to express the level of preference of each candidate independently of the others. The special case of two quality levels is called *approval voting*. No concessions have to be made by the voters to not waste their vote (*tactical voting*). Candidates are

elected with respect to their ranking by their average level of preference. While cardinal voting is not very popular for political elections, it is a often used in the context of sports competition, evaluations or TV shows. Voting protocols supporting cardinal voting must support ballots expressing a quality vector of size n.

2.3.2 *Proportional Systems*

Proportional systems serve to elect multiple winners, e. g. a board of representatives. Voters do not express their preference for the candidates directly, but for predefined party *electoral lists* of (ordered) candidates for a given party.

A formula is employed to determine the number of seats per list proportional to the number of voters. Only few votes are wasted due to rounding to integer numbers of seats and potential minimum quota.

2.3.2.1 *List Proportional Representation (List PR)*

The *list proportional representation system* is according to (Reynolds, Reilly and Ellis, 2005) the most popular voting system for the legislative body with about 70 countries out of 200 considered countries. Voters express their preference for one list out of many. The number of seats per list are determined (with a round-off error) in proportion to the received votes. The round-off formula can be a crucial factor and is part of the individual voting setup.

2.3.2.2 *Single Transferable Vote (STV)*

The *single transferable voting* has received strong support from political scientists, but its actual adoption is so far very low (Reynolds, Reilly and Ellis, 2005, p. 71). Voters rank all candidates comparable to the one-winner system *alternative vote*. However, it is generally not sufficient to get a majority of votes to get elected. The necessary threshold depends on the total number of votes and the number of seats.

$$\text{threshold} = \frac{\text{votes}}{\text{seats} + 1} + 1$$

Every candidate over the threshold gets elected. If there are less candidates satisfying the threshold that seats, the least popular candidate is neglected and his or her votes are redistributed according to the voter's ranking. The process continues until as many candidates satisfy the threshold as there are seats. The STV system is used for elections in e. g. Australia, Malta and Ireland.

2.3.3 *Mixed Systems*

When both plurality voting and proportional representation systems are used, but separately from each other, one speaks of *parallel systems*. Systems that use elements of both at the same time, are referred to as *mixed systems*.

2.3.3.1 *Mixed Member Proportional (MMP)*

The implementations of the *mixed member proportional voting* system differ throughout the countries that use this system, e. g. Germany, Canada and Italy. In some instances, voters express their preference separately for a candidate of their so-called single-member district (personal vote) and for a list (party vote).

For each single-member district, candidates are elected with the FPTP system applied to personal vote. To achieve a proportional representation with respect to the party vote, additional seats are added and filled with candidates from the respective list.

2.3.4 *Lottery Voting in Medieval Venice*

The former voting systems are all deterministic. For identical sets of voter preferences, the voting outcome is reproduced in every voting. However, voting systems can be found that incorporate randomness. The government head of the medieval Republic of Venice (13th to 18th century) was chosen in a complex process involving 10 passes (Coggins and Perali, 1998; Mowbray and Gollmann, 2007) comprising selection by lot and election by approval voting. While the former was employed to restrain corruption, the latter provided a safeguard to negate a leading drawback of the lot, the possibility to elect an unfit candidate.

The lottery voting serves as a conventional voting system that puts an interesting challenge to online voting protocols. While online voting protocols may easily allow for an approval voting, the implementation of a lottery that satisfies the requirements of correctness and verifiability warrants closer inspection.

2.4 PAPER-BASED VOTING

Paper-based voting protocols offer an unmatched solution to satisfy the correctness, privacy and verifiability to a high degree if properly deployed. According to *Comparative Data* (2014), more than 200 countries and territories rely on paper-based voting to elect the legislature.

Even though the particular procedure can vary slightly from country to country, there is a consensus on the principle aspects that are described schematically in the following to show how the basic security

properties are implemented (Election Process Advisory Commission, 2007; *Federal Electoral Regulations* 2017).

2.4.1 *Preparation Phase*

At the beginning of every voting, the eligibility requirements must be defined and individuals entitled to vote must be informed of the upcoming voting. Some countries require voters to register ahead of the voting while other countries register voters automatically based on their declared residency. Eventually, a register called *electoral roll* is produced that lists all eligible voters.

In order to cope with large numbers of voters, voters are partitioned to a reasonable number of voting districts by their residency. On voting day, the ballot casting and tallying is carried out in parallel and independently of each other in the *voting station* of every voting district.

2.4.2 *Casting Phase*

The voting station is the dedicated location to receive voters on voting day and is run by volunteering voters that become thus occasionally voting officers. The station is provided with undistinguishable ballot cards and the electoral roll of the voting district.

Voters have to present a proof of identity in order to cast their vote. If the voter is on the electoral roll and therefore confirmed to be eligible, a ballot card is handed out and the list is annotated accordingly to prevent that the same person will receive another. Under public surveillance, the voter alone will step behind a *voting booth* to fill and fold the ballot card in secret. The ballot card shall not be manipulated in a way suited to identify the card among other cards expressing the identical preferences. Further, no copy of the ballot card suited as a proof of the expressed preference shall be produced. Otherwise, the ballot card must be destroyed under public supervision and the voter may be provided a new ballot card in order to fill and fold anew his or her ballot card.

The filled ballot card is then thrown into the transparent ballot box. The transparent box renders ballot stuffing or false bottoms obvious.

2.4.3 *Aggregation and Evaluation Phase*

To ensure fairness, the aggregation can only start after the casting phase is terminated and no further votes can be cast. Under public supervision, the ballot box is emptied and votes are tabulated by the voting officers and potentially third parties. The aggregation result is published for each voting district.

Once, the results from all voting districts have been published, the voting outcome is determined with respect to the voting system chosen for the particular voting.

2.4.4 *Verification Phase*

The casting and aggregation phase is carried out in public. That means the protocol compliance can be verified by any other voter or observer, ideally, by the mutually distrusting representatives of the candidates. The verification of paper-based voting does not require any expert knowledge. Due to the transparent ballot box, ballot cards cannot be covertly manipulated, removed or added. The eligibility of other voters can be verified by the electoral roll that is amended publicly when voters receive their ballot card. The aggregation of all ballot cards can be observed. Hence, all aspects of the *correctness* of the voting can be verified.

The filling of the ballot in secret is enforced by public supervision. The folded ballot is cast into the ballot box and removed from the ballot box after shuffling only as one among many other ballot cards, so that no link between a voter and his or her ballot card can be established. Even if the ballot card is reproduced, e. g. in form of a smartphone photo, voters may voluntarily destroy their ballot and ask for a new one. In consequence, the production of a voting receipt becomes extremely difficult. *Privacy* is achieved by the uniformity of ballot cards and the human incapacity to track the individual ballots in the ballot box.

If the ballot cards are stored securely after the tallying, a recount in case of objections is possible.

The verification implements a *chain of custody*. The ballot cards can literally be followed by eye-sight which renders any malign or benign deviation from the protocol obvious. Voters do not have to reside to assertions by experts concerning the security of the voting, but can convince themselves. No trust in institutions or technology is assumed.

2.4.5 *Obsolescence of Paper-based Voting*

Even though paper-based voting provides an unmatched level of security, its system-wide properties, i. e. its flexibility, efficiency and voter convenience impose constraints in face of elections with several millions of voters and the recent advances in technologies.

Already in Ancient Greece (cf. Section 2.2.1), the secret ballot voting has only been employed in votings with a rather limited electorate of at most several hundred persons (Boegehold, 1963). For an electorate in the order of thousands of voters, the preference of the majority was mostly assessed by approximative open votings such as voting by shout or voting by the show of hands. More accurate and secret vot-

ings would not have allowed to conduct as many votings as scheduled during the assembly that requires the presence of the entire electorate.

Since then, agglomerations emerged that demand to cope with an electorate in the order of millions of voters. Only the introduction of many voting offices run in parallel allows to rely on paper-based voting. Otherwise, the casting phase would need to last for several days in order to give every voter the opportunity to participate. While the *parallelisation* is an important innovation, it is not without issues. Every voter can only supervise one voting station at a time. While candidates with large support may arrange for trusted observers in all stations, this does not hold true for candidates with few supporters. On the other hand, manipulations of global impact also require a larger coalition, because each voting office must be considered individually.

With its 12 standard times, France holds so far the world record for the country with the most time zones, beating even the United States and Russia, which have 11 time zones each.

The availability of advanced technology also poses a threat to the security of paper-based voting. The rapid dissemination of information on a global scale allows to publish preliminary voting outcome estimations that may be obtained from voter exit polls or by the analysis of social networks. This is especially an issue for votings in areas spanning different time zones in which the casting phase is bound to the local time. A prohibition by law is often in place, but not effective any more. The information can easily be distributed from within a different jurisdiction. Hence, the *fairness* is diminished. Note that also the analysis of social network data may allow for an increasingly precise estimate even before any vote has been cast (MacWilliams, 2015).

Moreover, the *privacy* of paper-based voting is at stake. Already cast ballot cards may appear at first sight to lack any feature suited to identify voters. In fact, ballot cards may be recognised by a mark of invisible ink, the voter's fingerprints or DNA traces that may subsequently be matched against nation-wide databases. Of course, the analysis may be prohibited, but there is still the potential that malicious voting observers employ those techniques covertly, e. g. using special miniature cameras. Even though paper-based voting does not require expert knowledge for casting and tallying, it may be required in future to detect such sophisticated attacks.

In practice, also the *verification* is limited. Most voters do not afford the time to observe the entire casting and tallying procedure. Often, the verification is carried out by very few individuals in comparison to the size of the electorate. In the hypothetical event of a high demand to assist the supervision, voters would need to be rejected despite the granted provisions, because the available space from within the voting procedure can be observed by eye-sight is limited in practice. The universal verification is therefore little convenient and constrained to a limited number of voters.

2.5 ONLINE VOTING

Online voting, i. e. remote electronic voting, inherits the potential advantages of electronic voting:

- The aggregation of cast ballots may be carried out rapidly and efficiently due to automation. Even votings with multiple races and complex voting systems may be handled with ease.
- The correctness may be improved due to the lack of human mistakes during the aggregation.
- Voters may receive an immediate feedback to confirm the recorded voting preference and to prevent voting unintentionally invalid.

Furthermore, online voting allows voters to cast their electronic ballot remotely via a network, e. g. the internet, which greatly enhances voter convenience. Voters are no longer required to cast their vote in a particular voting station where queuing might be necessary during rush hours. Instead, voters may choose at their own discretion where and at which time they cast their ballots.

A downside of all protocols with electronic ballots is the verification by voters with no expert knowledge. Unlike paper ballots, electronic ballots cannot be examined by eye-sight so that the *chain of custody* cannot be verified.

In case of online voting, the remote casting location does does not allow for a public supervision of the casting, so that privacy cannot be publicly enforced.

2.5.1 *Public Debate*

Many renowned voting security researchers expressed their concerns towards the security of online voting (Simons, 2004; U.S. Public Policy Council, 2010; Dill et al., 2012; Simons and Jones, 2012; Horwitz, 2016; Schneier, 2016). However, online voting is being promoted in many countries by the software industry, election officials and well-meaning people who underestimate the challenges to implement secure online voting (cf. Section 2.1). That being said, their arguments focus mainly on efficiency and convenience and the supposed consequences.

Supporters of online voting argue that votings could be carried out more cost-efficiently. Paper-based voting requires a committee of voting officials in each voting station during the casting and aggregation phase. No ballot cards must be printed and later securely stored for a potential recount. However, the up-front costs of online voting can be steep, e. g. several million USD (Simons and Jones, 2012), and past propositions imposed most often license fees per vote.

While many western democracies experience a drop in voter turnout, supporters wish to increase the voter convenience to allow more voters to cast their ballot. While there are voters who cannot cast their ballot due to other commitments (Colmar Brunton Social Research Agency, 2012), there is no significant evidence that the adoption of online voting has a significant positive impact on voter turnout (Kitsing, 2011).

2.5.2 *Selected Trials*

The risks and benefits of online voting have been assessed by governments of several countries. The derived policy and the adoption of online voting of four countries with largely diverging conclusions shall be shortly reviewed in the following to illustrate the different prioritisations of voting protocol properties. An extensive presentation is given in (Driza Maurer and Barrat, 2015).

2.5.2.1 *Case of Germany*

While online voting has never been available to the general public for legally binding elections in Germany, electronic voting trials were carried out from 1998 to 2008 in some cities.

The principles of general, direct, free, equal and secret general elections are incorporated in the German Basic Law (*German Basic Law* 2014) since 1949. Since then, election procedures were reformed several times (Volkamer, 2010) to account for the increased mobility of the voters. Voters were allowed to cast their vote per letter (*postal voting*) or before the official casting phase (*early voting*).

Direct-recording electronic (DRE) voting devices have been used during the general election in 2005 by about 2 million voters corresponding to about 3 % of the electorate. Further, the city of Hamburg was investigating the use of a DRE voting protocol with paper trail for its municipal elections in 2008 to cope with the increased aggregation complexity after the voting system had been changed to cardinal voting. Eventually, the prototype was not employed due to negative press and security reservations. The aggregation by hand lasted for four days and the entire voting caused costs of about 12 million Euros.

Public authorities have been pretty much in favour for DRE voting devices and turned down a) a petition at the Federal Parliament against the application of insecure and in-transparent voting computers and later b) a filed protest at the scrutiny committee of the Federal Parliament arguing that the advantages of such systems would prevail and alleged security issues 'obviously causeless'. The claim was filed at the Federal Constitutional Court and accepted for a hearing in 2008. The Court ruled in 2009 that the employed DRE voting devices are unconstitutional, because it must be ensured that 'essential steps of the voting and of the determination of the result can be examined by the citizen

reliably and without any special knowledge of the subject'. The public verification cannot be compensated by official institutions asserting the proper functioning of voting devices based on the examination of sample devices ahead of the election.

The Court did not prohibit DRE or online voting entirely. If the 'public nature of elections', i. e. the verifiability, is not compromised, or other constitutional interests justify exceptions, voting devices may be employed for general elections.

2.5.2.2 *Case of France*

France passed a law in 1969 to allow the use of mechanical voting devices for general elections in order to reduce fraud. While those devices were abandoned due to missing improvements, the same law was amended in 2003 to enable DRE voting. Previously, pilot programs in parallel to the 2002 elections were carried out. On those grounds, policy advisers recommended to enable DRE voting in voting stations for citizens in France and online voting for citizens residing out of France (Collard and Fabre, 2014).

The adoption of electronic voting was controversially discussed. Eventually, only one municipality used compulsory DRE voting for the general elections in 2007. Despite of the reported anomalies in about 30 % of those voting stations, the Constitutional Council affirmed the use of DRE voting. Different reports were published condemning the use of online voting due to the lack of verifiability. Nonetheless, citizens abroad could choose among other methods to cast their ballot online for the general elections in 2012. In 2017, the French authorities decided to suspend online voting due to increased concerns of election hacking.

2.5.2.3 *Case of Estonia*

Estonia, a small country in Northern Europe, is a pioneer in e-governance. Its citizens are well known to adopt online services early. They use online banking since 1996, submit tax declarations online since 2000 and can access a variety of services using their digital identity cards that allows for legally binding digital signatures. So it came at no surprise when online voting was first considered by the government in 2001 and in 2002, a corresponding law was passed (Kitsing, 2011).

Estonia became in 2005 the first country that provided online voting for legally binding general elections (Ülle and Martens, 2006). In the general elections in 2015, 30 % of all votes were cast online. So far, Estonia is supposed to be the only country with a considerable fraction of online voters.

Even though online voting is popular, it is reported to be vulnerable against covert client side attacks and against server side attacks due to improper execution or insecure procedural routines. The employed

system faced heavy criticism due to its limited means of verification (Springall et al., 2014). Authorities plan to improve the system for coming elections with the goal to provide cryptographic proofs that votes were counted as cast and cast as intended.

2.5.2.4 *Case of Australia*

Australia is a country with a large territory sparsely populated. Voting in general elections is compulsory. Therefore it is assumed important to provide solutions for those voters who are for legitimate reasons unable to attend on voting day the voting in person. While postal voting and later early voting were for long-time the only alternatives.

DRE voting with and without audit trail was introduced in various regions in the 2000s years that allowed more disabled voters to cast their vote autonomously.

Online voting was first piloted in 2007, but restricted to Defence employees. Due to a court decision in favour of the blind and low-vision community, online voting via internet or telephone was made available for the New South Wales general elections in 2011. Eligible for online voting were only voters with disabilities or those in transit. An improved online voting system was also offered in the 2015 elections (Brightwell et al., 2015) and 280 000 voters cast their vote online, which constitutes so far the largest legally binding online election. Experts advised against the adoption of online voting and presented serious security flaws of the deployed system (Halderman and Teague, 2015). Regardless of these findings, New South Wales plans to continue online voting.

2.6 SCOPE OF APPLICATIONS

Voting is the main privacy preserving aggregation realised with predigital technologies. It is covered extensively in the next chapters. Though, there have been other kinds of aggregations with similar requirements, such as lotteries and auctions. Their implementation may also take advantage from advances in voting technology.

During the last years, many services emerged that rely on the evaluation of large datasets of inter alia personal data, i. e. Amazon's buy recommendations, Google's website ranking or Spotify's song similarities. A privacy preserving aggregation may improve the confidentiality of the data providers.

2.6.1 *Lottery*

Lottery is a form of gambling in which a reward sponsored by a set of players is randomly redistributed to a subset of those players. The random process used to determine the winners, e. g. by the drawing of

lots, is crucial for a fair lottery. In the past, lotteries have been proven to be very vulnerable to manipulations, which is why countries are prohibiting them, or applying strict regulations or allowing only for state-run lotteries.

A very simple lottery protocol consists of multiple players that contribute to the reward in equal shares and receive in return a lottery ticket that is placed in a ballot box. Eventually, the authority draws one random ticket from the ballot box and its corresponding player wins the reward. Only eligible players must place tickets in the box. The players can verify the correctness of the random process by personal supervision throughout the drawing. To prevent the intimidation of the winner, e. g. in form of begging or coercion, the identity of the winner shall be confidential. Similar to voting, lottery protocols must ensure correctness and privacy in a verifiable manner.

While online lotteries become more popular for their greater convenience, players have most often no means of verifications and have to trust the assertions of authorities. Online voting security concepts can improve the verifiability of online lotteries. The case is studied in Chapter 7.

'Random numbers should not be generated with a method chosen at random.'
—Donald Knuth

2.6.2 *Auctions*

An auction is a procedure to buy and sell goods. Given a group of multiple potential buyers, the purchase shall be offered to the buyer that is willing to pay the most in a situation in which the willingness is a priori confidential.

Very common is the *English auction*. An auctioneer announces publicly a minimum offer that is then raised in turns. Buyers can reveal in each turn their willingness to buy by placing a *bid*. The purchase is concluded for the largest offer that received exactly one bid.

Less common is the *Dutch auction*, in which the auctioneer announces decreasing offers from a high starting price. The purchase is concluded with the first bid. This second form does not reveal the willingness of other potential buyers and provides therefore more confidentiality.

Confidentiality is in both cases important. If in English auctions, the auctioneer learns the accurate willingness of every potential buyer, the purchase may become more expensive than just the first offer receiving only one bid. In the Dutch auction, the purchase may become less expensive if the buyer knows the willingness of all other potential buyers. Furthermore, the auctioneer and all buyers must be provided means to verify the correctness. Bids can only be placed during the bidding phase and no bids can be withdrawn.

2.6.3 *Aggregation of Sensitive Data*

Various innovative services that issue predictions or recommendations rely on the analysis of large datasets. For example, services based on machine learning algorithms must be trained with samples in order to produce reliable results.

If the dataset contains personal data, special measures must be taken to ensure confidentiality of the data sources which may be required by law. Consider for example the aggregation of medical patient data in hospitals for research. Similar to votings, there is an interest to compute accurate global indicators while retaining the confidentiality of the data sources. This problem has already been addressed and current solutions comprise secure multi-party computation and secret sharing schemes. Their limitations are addressed in the following chapter.

3

STATE OF THE ART

When returning the ballot through the secure online delivery system, you are voluntarily waving [sic] your right to a secret ballot and are assuming the risk that a faulty transmission may occur.

— Alaska State Division of Elections
(Horwitz, 2016)

Paper-based voting relies on two principles. First, the set of all cast ballots in the ballot box can be observed throughout the voting to assert its *integrity* by a *chain of custody*. By the observation of the ballot transport, the integrity of the voting can be determined. Second, voters are identified to proof their eligibility and cast subsequently their look-alike ballot into the box. A cast ballot cannot be distinguished without more ado from any other ballot in the box. Even with careful observation, the human capacity to trace one ballot among many in the box is exceeded. *Privacy* is achieved. From pebbles ballots employed in Athens in the 5th century (cf. Section 2.2.1) to paper-based ballots nowadays, these two principles have been maintained.

The adoption of ballots, which cannot be captured by the virtue of the human senses, or virtual ballots with no physical presence is inherently incompatible with the approach of the *chain of custody*. Electronic ballots offer great potential to improve automation and mobility. They can easily be cloned or altered to allow for transmission via wire or over air and for data processing in the course of the ballot aggregation. Obviously, those properties are in stark contrast to the idea to maintain a chain of custody.

Consequently, an alternative must be provided for the verification of electronic ballots that rely neither on ballot observation nor on the integrity of the ballot transport. Various concepts have been considered and are reviewed in the following Section 3.1. Nonetheless, the overwhelming majority of current online voting protocols, as we detail hereafter, are either lacking properties, so that trust in authorities carrying out essential tasks must be assumed, or use advanced cryptography, which imposes trust in technology as expert knowledge cannot be implied.

In a situation where trust in authorities or in technology seems to be unavoidable, it is a good practice to a) let the voters freely choose their trustees or b) share the authority among as many equipotent individuals as possible, a concept commonly known as *separation of powers*. If the authority is shared among all stakeholders, i. e. all voters in the case of a voting, the protocol is essentially distributed. Section 3.2 is dedicated

to review distributed protocols with promising concepts for online voting. The chapter concludes with a taxonomy of protocols with respect to their separation of powers and trust assumptions in Section 3.3 and a summary in Section 3.4.

3.1 SECURE ONLINE AGGREGATION

Different protocols have been proposed to realise a secure online aggregation of which some address all phases of a voting while others provide only building blocks. The primary focus for this work shall be their assumptions and limitations with regard to verifiability and the subsequently assumed trust in authorities or technology. Therefore, we employ the concepts of *end-to-end* (e2e) verifiability (Aleksander Essex, 2013; Ali and Murray, 2016) and *software independence* (Rivest, 2008).

END-TO-END VERIFIABILITY A voting protocol is end-to-end verifiable if compelling evidence can be provided to convince individual voters or their chosen trustees that their votes have been tallied as intended. The evidence must cover the entire cycle of the ballots from the casting to the announcement of the tally and assert that ballots are:

a) cast as intended,

b) recorded as cast, and

c) tallied as recorded.

SOFTWARE INDEPENDENCE A voting protocol is software independent if an (undetected) change or error in its software cannot cause an undetectable change or error in an election outcome. One distinguishes further:

a) Protocols allowing for a *recovery* without rerunning the voting in cases of changes or errors are called *strongly software independent*.

b) Protocols that do not offer *recovery* and require to rerun the voting in cases of changes or errors are called *weakly software independent*.

Software independence can contribute to ensure e2e verifiability. The design of electronic voting protocols providing both privacy and correctness is a complex matter that lead most often to complex software. Generally, it is in practice either impossible or highly demanding and extremely expensive to write software with no errors. Software may be without errors, but then it is still effectively impossible to provide evidence for it (Rivest, 2008). Hence, software dependent voting protocols break the chain of evidence and are not e2e verifiable. Bernhard

et al. (2017) provide an overview on essential security properties and corresponding open questions.

The following high-level overview of secret aggregation concepts is based on the surveys (Lambrinoudakis et al., 2003; Jonker, Sjouke Mauw and Pang, 2013).

3.1.1 *Trusted Authorities*

A trivial online voting protocol assumes that all voters agree on a common *trusted authority*. Further, the existence of pairwise secure channels between the authority and each voter is assumed, which may be realised using e. g. pre-shared keys and communication security protocols such as transport layer security (TLS).

During the casting phase, voters send their ballot via the secure channel to the authority. The authority is entrusted to keep ballots confidential, to tally all ballots and announce the final tally. The authority learns the vote of every voter. Confidentiality and correctness can only be verified by the authority itself.

While this protocol is certainly not suited for high stake votings such as political elections due to its demanding assumptions rarely met in practice, it is wide-spread in e. g. television shows and various internet services such as Doodle or Google Forms.

The assumptions can be slightly weakened by the separation of powers from one authority to a set of a few authorities (Lambrinoudakis et al., 2003). However, the fundamental issue of the concentration of power to just few individuals persists.

3.1.2 *Anonymous Voting*

Similar to paper-based voting in a polling station, *anonymous voting* employs ballots that cannot be linked to the voter. Consequently, a distinct proof of eligibility must be provided to prevent ballot stuffing. Hence, voters acquire in a first step from the authority an *eligibility token*. Therefore, voters must be authenticated. In the second step, voters cast their ballot over an anonymous channel and provide along with their ballot their eligibility token. Such a token must not reveal the voter identity.

3.1.2.1 *FOO Online Voting*

The FOO online voting protocol named after their authors Fujioka, Okamoto and Ohta (1993) employs blind signatures (Chaum, 1983) to generate eligibility tokens. It is presented hereafter in a simplified version without prior ballot commitments to demonstrate the principle concept of anonymous voting.

The model consists of an authority A, n voters P_i with $i = 1, \dots, n$, a public key infrastructure (PKI), pairwise authenticated and anony-

mous channels between A and each P_i, and a public bulletin board (PBB). The following notation adapted from the original work is employed:

A	Authority
P_i	Voter, i-th out of n
a_i	Vote of P_i
$\sigma_i(m)$	P_i's signature scheme using its key pair (pk_i, sk_i)
$\sigma_A(m)$	Authority's signature scheme
$\chi(m, r)$	Blinding technique for message m and random number r
$\delta(s, r)$	Retrieving technique of blind signature

PREPARATION PHASE The voter P_i chooses its vote a_i and applies the blinding technique to get $a'_i = \chi(a_i, r_i)$ with r_i a random blinding factor. Then, P_i sends the blinded vote a'_i to the authority A via the authenticated channel. P_i provides further to A its signature on the blinded vote to prevent tempering and to allow A the verification of its eligibility. If P_i is eligible and has not voted yet, A computes the blinded vote signature $s'_i = \sigma_A(a'_i)$ and sends it to P_i.

At the end of the phase, A publishes on the PBB a signed list containing all received blinded votes a'_i, their signatures of P_i and the identity of P_i.

CASTING PHASE P_i computes its eligibility token $s_i = \delta(s'_i, r_i)$ and checks the signatures using A's public key. Then, P_i sends the vote a_i in plain text and its eligibility token s_i to A using the anonymous channel.

AGGREGATION PHASE A publishes on the PBB the received votes a_i along with their tokens s_i. Everyone can compute the tally and subsequently the voting outcome.

The original protocol FOO can be considered as a building block for online voting and has been discussed and extended numerous times (S. Mauw, Verschuren and Vink, 2007; Jonker, Sjouke Mauw and Pang, 2013). It is strongly software-independent and provides e2e verifiability. The privacy is reduced due to ballot receipts in form of the ballot commitment key, or in the simplified version, the blinding factor r_i.

FOO is carried out in three rounds, respectively, two in the simplified version. A voting protocol requiring more than one round is often considered as impractical, because voters may abandon the voting halfway. Voters are assumed to have trust in the technology, in particular, in the blinding technique. However, voters may choose the software to prepare and verify electronic ballots, the so-called polster, at their own discretion. In consequence, implementation flaws are unlikely to occur on a global scale.

There are no public reports of recent deployments of FOO or similar protocols. In practice, verifiable anonymous channels are difficult to achieve. Moreover, the encryption of electronic ballots may be required by law as it is the case in numerous countries, e. g. France.

3.1.3 *Random Perturbation*

In paper-based voting, the link between a cast ballot and the corresponding voter is broken, because ballots are shuffled in the ballot box and get out in an essentially random order. Obviously, the shuffling does not harm the integrity of the ballots.

Random perturbation transfers this concept to electronic ballots. Voters send encrypted ballots to a group of n mutually distrusting authorities that shuffle one after each other the set of all ballots. Shuffling can be realised using Mix-Nets (Chaum, 1981). Afterwards, ballots are decrypted to compute the tally. To prevent the malicious decryption of ballots before the shuffling, a (k, n)-threshold cryptography can be employed (Pedersen, 1991), which requires the cooperation of k out of n authorities. Ballots cannot be linked to voters if at least one authority is honest. All authorities would need to collude in order to reveal the ballot of a voter.

To achieve e2e verifiability and software-independence, voters must be provided evidence that authorities have not removed, altered or added ballots during the shuffling and decryption. Cryptographic proofs such as zero-knowledge proofs may be employed (Adida, 2008).

The employed cryptography is very complex and without expert knowledge, voters rely on trust in technology. Further it is assumed, that every voter has trust in at least one authority. This is problematic, because the number of authorities is usually very constrained to keep the shuffling and decryption procedure efficient which exhibit at best a computational complexity that is polynomial in the number of ballots and authorities.

3.1.4 *Homomorphic Encryption*

Both anonymous voting and random perturbation provide means to break the link between the voter and its (decrypted) ballot, so that only anonymous ballots are aggregated. In the *homomorphic encryption* approach, it is not the identity of the voter which is concealed, but the vote itself. Encrypted ballots are aggregated and only the final tally is eventually decrypted. Therefore, a cryptosystem denoted $\mathrm{enc}(\cdot)$ that provides the homomorphic encryption property is employed, i. e. it holds for messages m_1, m_2, message aggregation $\oplus$ and cyphertext aggregation $\otimes$:

$$\mathrm{enc}(m_1) \otimes \mathrm{enc}(m_2) = \mathrm{enc}(m_1 \oplus m_2)$$

(k, n)-threshold cryptography is employed to prevent the decryption of individual ballots before their aggregation. Zero-knowledge proofs ensure that the encrypted ballot represents a valid vote.

Note that not all voting systems (cf. Section 2.3) are readily compatible with homomorphic encryption. Those based on sums such as FPTP or approval voting are easy to implement. Ranked voting or more complicated systems may be impractical, because of the large number of $O(n!)$ distinct choices for n options to rank, which must be separately counted.

3.1.4.1 *Helios Online Voting*

The well-studied online voting software Helios (Adida, 2008; Chang-Fong and Essex, 2016) is based on homomorphic encryption since version 2 (Adida et al., 2009). It has been used so far for internal elections in universities and scientific communities.

The model of Helios consists of a set of authorities, voters and optional auditors. Pairwise secure and anonymous channel is assumed between the authority and each voter. A PBB allows to publish information for the verification.

PREPARATION PHASE Using (k, n)-threshold cryptography, the authorities create each their share of the ballot decryption key and the public key for encryption. Voters and auditors may create as many electronic ballots as they wish. Therefore, ballots are filled and encrypted using the public key. A non-interactive zero-knowledge proof ensures the validity of the ballot. Then, the encrypted ballot and the proof are sent anonymously to the authority that commits to the encrypted ballot and provides subsequently a confirmation. The commitment may be challenged. For this, the authority must provide the used secret to proof its honesty. Both the ballot and the commitment are marked as invalid and a new ballot must be created.

CASTING PHASE To cast a ballot, now authenticated voters ask to seal a ballot with an unchallenged commitment. The authority stores the name of the voter and its ballot on the PBB for public verification. Voters may cast repeatedly. Only the latest ballot is counted then.

AGGREGATION PHASE The encrypted tally is computed from all valid encrypted ballots by homomorphic addition and jointly decrypted by a set of authorities meeting the threshold of the decryption scheme.

3.1.4.2 *Discussion*

Helios is advertised as as open-audit protocol. It is software-independent and e2e verifiable by voters and auditors. The issue of coercion inherent to online voting has been addressed, even though not entirely

resolved. Voters may change their vote throughout the casting phase which defends most cases of over-the-shoulder coercion.

Voters are assumed to have trust in technology and in sufficiently many authorities, so that the threshold cannot be met by a malicious coalition. As before, the number of authorities is constrained by the scalability of the distributed key generation scheme. The PBB of Helios is implemented as a single web server and must be trusted to provide a consistent view to all voters.

3.1.5 *Secret Sharing*

In online voting protocols achieving secrecy by the encryption of the ballots, (k, n)-threshold cryptography has been employed to raise the robustness of the protocol. With cryptographic keys shared among multiple authorities, the corruption of a single individual does not suffice any more to break the secrecy. Instead, k at best mutually distrusting authorities must cooperate.

This approach can be applied to protect the secrecy of the ballots. *Threeballot* is a paper-based voting protocol that demonstrates how ballots can be split into shares (Rivest and Smith, 2007). Here, voters fill each three identical complete ballots that constitute one multiballot. To approve one candidate, it must be approved on exactly two ballots. Respectively, to disapprove, a candidate must be approved on exactly one ballot. The integrity of the multiballot is verified by a trusted authority before ballots are dropped into the ballot box. Subsequently, the ballots of one multiballot cannot be identified anymore.

The *secret sharing scheme* (Shamir, 1979) provides for a set of tallying authorities, that each receive one share, compute intermediate tallies and then jointly the global tally. Without cryptography, the integrity of shares cannot be verified and malicious voters may provide shares that constitute ballots of increased weight to tamper with the voting outcome. While verifiable secret sharing schemes have been explored (Chor et al., 1985; Rabin and Ben-Or, 1989), their capacity to ensure the validity of shares is limited and employ complex cryptography. Another direction has been chosen for *DPol*, that achieves probabilistic verification due to the introduction of intermediate tallies.

3.1.5.1 *DPol Decentralised Online Polling*

The approach of *DPol* (Guerraoui et al., 2012) consists of a partition of voters into clusters. Voters belonging to one cluster casts their ballot shares into a local ballot box, which is administrated by voters of another cluster. It is assumed that the majority of the voters is honest and follows strictly the protocol. Dishonest voters may not harm their own reputation due to manipulations that are revealed with certainty. In this adversary model, DPol demonstrates that probabilistic correct-

ness and probabilistic privacy can be accomplished on the basis of the secret sharing scheme. In consequence, no cryptography is employed to provide secrecy and correctness.

The model of DPol consists of n voters partitioned into $\sqrt{n}$ clusters of $\sqrt{n}$ voters each. The clusters are aligned on a ring, so that each cluster has one preceding and one succeeding cluster (Figure 3.2 (a)). Every voter gets assigned $2k+1$ recipient voters from the succeeding cluster on the ring and is itself recipient to $2k+1$ voters from the preceding cluster. k is a privacy parameter. Secure channels between the voters in one cluster and voters and their recipients are assumed. DPol considers FPTP voting with d candidates.

PREPARATION PHASE The ballot for the i-th candidate is given by the vector $\vec{b} = k\sum_j \vec{e}_j + \vec{e}_i$ with the orthonormal basis vectors $\vec{e}_j$. The ballot $\vec{b}$ is decomposed in $2k+1$ shares $\vec{b}_1, \ldots, \vec{b}_{2k+1} \in \{0,1\}^d$ that contain each at least one 1 and one 0. For example, to promote $i = 1$ with $d = 3$ and $k = 1$, the ballot could be decomposed as $\vec{b} = (2,1,1) = (1,0,1) + (0,1,0) + (1,0,0)$.

CASTING PHASE Once a voter has generated its $2k+1$ ballot shares, it sends each of them to a different recipient. Every voter is itself recipient for $2k+1$ voters and receives of each of them one share.

AGGREGATION PHASE Every voter computes the individual tally by summing up the received shares. The individual tallies are exchanged with all other voters of the same cluster in order to compute the tally for the preceding cluster. The tallies are then forwarded around the circle until every voter knows the tally of all clusters to compute eventually the global tally.

The integrity is based on the verification of valid shares by the majority of honest voters. More details are provided in the original work (Guerraoui et al., 2012).

3.1.5.2 *Discussion*

Few extensions have been proposed, among them EPol (Hoang and Imine, 2015), that generalise DPol to other network topologies than circles and allow for more flexible numbers of voters n than square numbers.

DPol and EPol do not meet the security requirements and are therefore not online voting protocols in the classical sense. The assumption of a honest majority and attributes that hold only with a given probability are highly non-conventional for online voting protocols. Still, DPol can be considered as a land-mark in online voting technology. Voters are equipotent and carry out the aggregation in a distributed manner. There is no trusted authority and as such no software runs on a central

server. In this particular setup, software independence and e2e verifiability are not well-defined (cf. Section 3.4).

3.1.6 *Secure Multi-Party Computation*

The aim of secure multi-party computation (SMC) is to compute collectively a joint function over the secret inputs of all participants. The correctness and secrecy properties rely on cryptography (Rabin and Ben-Or, 1989). SMC seems to be an appropriate technique to compute a tally from secret ballots (Yao, 1982). However, several issues render an implementation of a voting protocol difficult. As (Gambs et al., 2011) points out, the communication complexity for n voters is $O\left(n^2\right)$, or in the case of the existence of a trusted party, $O\left(n\right)$. Consequently, SMC is impractical to conduct large-scale online votings with no trusted authority, though, it may be suited for boardroom voting protocols with only few voters. One proposition to realise an online aggregation based on SMC is to partition voters into clusters and run a SMC protocol in every cluster with only few voters each. The intermediate results are then tallied by other means to get the global tally.

3.1.6.1 *SSP Scalable and Secure Online Polling*

The online polling protocol *SPP* (Gambs et al., 2011) employs SMC based on homomorphic encryption and zero-knowledge proofs to compute the intermediate tallies for subsets of the electorate with n voters in total. Then, the intermediate tallies are aggregated along a tree structure. This way, the communication complexity can be improved from $O\left(n^2\right)$ to $O\left(n \log^3 n\right)$.

PREPARATION PHASE Voters are randomly grouped using a chord overlay into clusters of equal size in order to partition potential dishonest voters. The clusters are then arranged in a binary tree structure (Figure 3.2 (b)) to pre-determined nodes.

Voters in the root node cluster jointly create a (k, n)-threshold decryption key for homomorphic cryptography. While the voters in the root node cluster keep their share of the secret key private, they communicate the public key to all other voters.

CASTING PHASE In each cluster, voters determine jointly the tally of the cluster using a SMC based on homomorphic cryptography and non-interactive zero-knowledge-proofs to back the validity of their votes. Using a broadcast channel connecting all voters of a cluster, votes with proofs are announced within the cluster. Taking advantage of the homomorphic property, all encrypted ballots are tallied by every voter in the cluster.

AGGREGATION PHASE If the tree node of the cluster has child nodes, voters wait to receive the partial tally of the respective subtrees. Once, the subtree tallies have been confirmed by a majority of voters of the cluster of the corresponding child nodes, the subtree tallies are added to the subtree tally of the voter's cluster. If there is a parent cluster, the subtree tally is forwarded to its voters using a secure channel. The procedure is repeated until the voters in the root clusters have eventually computed the tally of the entire tree. A set of voters in the root cluster, that meets the required threshold, jointly decrypts the global tally that is then subsequently forwarded along the tree to all voters. A non-interactive zero-knowledge proof allows to verify that the global tally has been faithfully decrypted.

3.1.6.2 *Discussion*

As the protocol DPol, SPP carries out the aggregation with no trusted authority. However, few random voters are assigned the special role to provide encryption keys at the beginning and decrypt the global tally at the end. It is assumed that those voters are trusted to the extend, that no malicious coalition meets the threshold to decrypt individual ballots or intermediate tallies. Voters rely on other voters to ensure the correctness of the intermediate tallies in their clusters.

3.2 DISTRIBUTED PROTOCOLS

Distributed protocols provide procedures allowing autonomous systems, in this context called peers, to jointly execute and coordinate their concurrent actions by message passing in order to achieve a common goal. In comparison, to their centralised counterparts, distributed protocols promise a greater fault-tolerance and scalability as we show hereafter. Major P2P networks must cope with millions of users. Already from the 1990s, decentralised voting was considered in the domain of robust data replication and distributed consensus, but proposals were not addressing privacy yet (Nakajima, 1993; Lamport, 1998).

Protocols such as DPol and SPP pave the way for privacy preserving online voting protocols that are distributed. Privacy and correctness are balanced locally during the in-network aggregation. This motivates a review of further distributed protocols that demonstrated already their capacity to deliver robust and verifiable services to a large number of participants and contribute concepts for novel distributed online voting protocols.

3.2.1 *Distributed Hash Tables*

The coordination of peers in a distributed protocol relies on the passing of messages among peers. In a trivial approach, one authority keeps

track of all peers and their current state and routes peer-to-peer (P2P) messages. This model is employed by many protocols such as load balancing or internet relay chats. Here, the authority constitutes a single point of failure. The processes come to halt if the authority ceases to function. Further, the authority limits the scalability of the distributed protocol.

Server issues lead to a two-day global outage of Skype in 2007. A DDoS attack caused in 2016 an hours-long outage of Twitter, Amazon, Netflix and other sites (Thielman and Johnston, 2016).

Solutions have been developed to distribute the discovery of peers and the routing of messages among peers to improve the scalability and robustness of the protocol. Among them are distributed hash tables (DHTs) which provide a replicated database to store in the memory of its peers arbitrary key-value pairs, and foremost information concerning the availability of peers and message routing (Androutsellis-Theotokis and Spinellis, 2004). Typical operations are lookups for peers or values for given keys and the storage of values.

3.2.1.1 *Kademlia*

A well-established DHT is *Kademlia* (Maymounkov and Mazieres, 2002) that is used by large file-sharing and botnet networks. Kademlia operates on a tree overlay. Its concepts are recalled hereafter. The notation is as follows:

P_i	Peer, i-th ouf of n
k	Maximum number of contacts per tree segment (k-bucket)
x	Kademlia leaf node ID (KID) of size B
B	Size of a KID in bits, e. g. 160
x_i	KID of peer P_i
d	Node depth, i. e. number of edges from the node to the root
$\eta(\cdot)$	Cryptographic hash function, e. g. SHA-3
$\widehat{\mathbb{S}}(x,d)$	Subtree whose root is at depth d which contains leaf node x
$\mathbb{S}(x,d)$	*Sibling subtree* whose root is a sibling of the root of $\widehat{\mathbb{S}}(x,d)$

Kademlia provides for a binary tree overlay network in which each peer P_i is assigned to a leaf node. The leaf node identifiers $x \in \{0,1\}^B$ with a size of B bits span the Kademlia binary tree of height B and are denoted KID. Each peer P_i joins the Kademlia overlay network using its KID x_i chosen uniformly at random from $\{0,1\}^B$. B is chosen sufficiently large, commonly 128 or 160, so that hash collisions leading to identical KIDs for distinct peers are very unlikely. Consequently, the occupation of the binary tree is very sparse.

Any node in the tree can be identified by its depth $d \in \{0,\dots,B\}$ and any of its descendant leaf nodes with KID x. A *subtree* $\widehat{\mathbb{S}}(x,d)$ is identified by the depth d of its root node and any of its leaf nodes x. The subtree notation is overloaded to designate as well the set of peers assigned to leaves of the corresponding subtree.

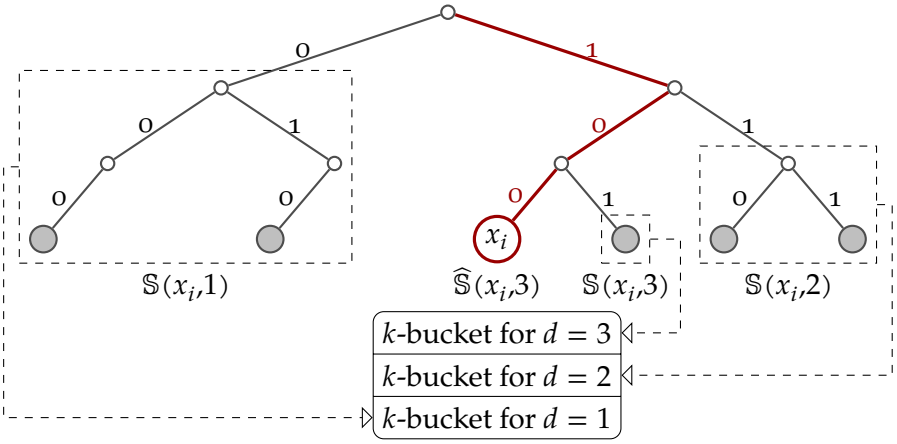


Figure 3.1: Example of Kademlia k-buckets. The sparse tree with $B = 3$ is partitioned for KID $x_i = 100$ into subtrees $\mathbb{S}(x_i, d)$ with root node at depth $d = 1, 2, 3$. The k-buckets for each d contain at most k peers $P_j \in \mathbb{S}(x_i, d)$.

Further, $\mathbb{S}(x, d)$ denotes the sibling subtree of $\widehat{\mathbb{S}}(x, d)$, so that $\widehat{\mathbb{S}}(x, d) = \widehat{\mathbb{S}}(x, d+1) \cup \mathbb{S}(x, d+1)$. The entire tree is denoted $\widehat{\mathbb{S}}(x, 0)$. One observes that $\forall d : P_i \in \widehat{\mathbb{S}}(x_i, d)$ and $\forall d : P_i \notin \mathbb{S}(x_i, d)$.

In Kademlia, the distance $d(x_i, x_j)$ between two KIDs $x_i = (x_i^1, \ldots, x_i^B)$ and $x_j = (x_j^1, \ldots, x_j^B)$ is defined as their bit-wise XOR ($\veebar$) interpreted as an integer.

$$d(x_i, x_j) = \sum_{\beta=1}^{B} (x_i^\beta \veebar x_j^\beta) \cdot 2^{B-\beta}$$

In general, a peer P_i with KID x_i stores information on peers with x_j that are close to x_i, i. e. for small $d(x_i, x_j)$. For this purpose, P_i disposes of a set denoted k-bucket of at most k players $P_j \in \mathbb{S}(x_i, d)$ for every $\mathbb{S}(x_i, d)$ with $d > 0$.[1] See Figure 3.1 for an example. The size of subtrees decreases exponentially for growing depth d. Consequently, the density of known peers of corresponding k-buckets grows exponentially.

The basic database operations are realised by remote procedure calls (RPCs) `STORE`(key, value) and `FIND_VALUE`(key) that allow to add, respectively, find entries. The RPCs `PING` to verify the availability of peers, and `FIND_NODE`(KID) to lookup peers are used to maintain the routing table, that is the set of all k-buckets. In order to connect to the Kademlia P2P network, one other connected peer must be known initially. The routing table is then populated by peer lookup RPCs for random KIDs to the closest already known peers. Those calls are responded with a set of closest, known peers from their routing table. One lookup might

1 Note that originally (Maymounkov and Mazieres, 2002) the common prefix length b is used to index k-buckets/sibling subtrees while we use the depth $d = b + 1$ of the root of the subtree.

require multiple, consecutive request-response cycles until the given peer has been found and might also fail if there is no peer (any more) with the given KID. `FIND_VALUE` RPCs behave basically like `FIND_NODE` for a KID $x = \eta(\text{key})$ in which requested peers that know the value for the given key reply with the value instead of a list of peers close to KID x.

The peer lookup time in a network of n peers is of $O(\log n)$ (Cai and Devroye, 2013). Both the connection to the network and the lookup of values is based on that RPC and therefore of the same order. The size of the routing table in memory is upper bounded by $O(B)$, because there are at most as many k-buckets as the height of the tree $h = B$. The k-bucket is limited to at most k peers who occupy each a constant amount of memory. By a probabilistic analysis to estimate the number of empty k-buckets, one finds that the average memory needs are of $O(\log n)$. The load of routing and value lookups is balanced using value replications to close peers to avoid bottlenecks due to overloaded peers.

A common issue in networks of autonomous peers are so-called *Sybil attacks*. Malicious peers may deliberately choose the identical KID of their target to receive its request in order to manipulate the response. An obvious solution is to introduce a trusted authority that assigns signed KIDs to peers. While the offered protection is perfect, the authority constitutes a bottleneck in the otherwise distributed protocol. A common scheme is based on a proof of KID ownership using signatures (Baumgart and Mies, 2007). Each peer generates its key pair (pk_i, sk_i) and derives its KID by hashing $x_i = \eta(pk_i)$. The ownership of x_i can be proven by a signature with the secret key sk_i of e. g. the RPCs and their responses.

3.2.1.2 *Discussion*

Kademlia provides a building block for efficient peer discovery in large peer-to-peer networks. No authority is required. The protocol is robust, because the load is balanced among all peers and dishonest peers cannot suppress the storage of values.

3.2.2 *File Sharing*

'Freedom of the press is limited to those who own one.'
—A. J. Liebling

The distribution of content in the internet using uni-directional, server-based protocols, for instance HTTP or FTP, place the transfer costs to a large extend on the server. The infrastructure and potential economical resources limit the scalability, and consequently, the coverage. Further, the distribution is vulnerable to disruptions of the server due to attacks or failures. Distributed content dissimination protocols such as *BitTorrent* covered hereafter, *IFPS* (Benet, 2014) or *Storj* (Wilkinson et

al., 2014) redistribute the transfer costs among the content audience, where transfer is often not metered.

3.2.2.1 *BitTorrent*

The BitTorrent file distribution protocol (Cohen, 2003, 2008) relies on a so-called *tracker* and peers that assume the role of a *leech* when receiving data and of a *seed* when sharing data. Peers may be *leechs* and *seeds* at the same time.

Initially, a single seed possesses the file to be distributed. Using a PBB, it provides file meta information, commonly as `*.torrent` file, containing the file name, file size, checksums and the choice of a tracker. Files are subdivided in small pieces of a pre-defined size. Peers that wish to receive or share the file, register themselves at the tracker. The tracker keeps track of all registered peers for a given file and provides on request a random subset of those. Peers contact the tracker in intervals to learn about other registered peers that are consequently requested to find out their available file pieces. Leechs download rare pieces first and may become a seed as soon as they are requested to provide an already downloaded piece.

A bottleneck in this protocol is the tracker, that imposes an upper bound on the number of peers it can handle. Without trackers, peers cannot discover each other. The protocol can be made more robust by allowing for more than one tracker to run in parallel. Further, the load of the tracker can be greatly reduced by a DHT such as Kademlia providing a distributed database of registered peers (Loewenstern, 2008).

3.2.2.2 *Discussion*

'BitTorrent is a free speech tool.'
—BitTorrent.org

BitTorrent provides great scalability and robustness based on cooperation. Assuming unselfish peers, the average number of downloads from every peer is of $O(1)$. BitTorrent is widely adopted to disseminate fast and efficiently large software updates and videos, synchronise databases etc. In the early 2000s, BitTorrent accounted for the majority of internet traffic (Cisco Systems, 2008), but has been outpaced since then by rich media streaming services provided by e. g. Youtube, Netflix or Spotify.

3.2.3 *Cryptocurrencies and Blockchains*

Electronic money transactions are, unlike electronic voting, well established and considered indispensable for today's global economy. The Society for Worldwide Interbank Financial Telecommunication (SWIFT) reports of a daily average of about 27 million transactions (*Traffic & Figures* 2017). Similar to electronic ballots, electronic money allows for easy transfer. Special care must be taken to prevent fraud such as double

spending or unauthorised transfers. The prevailing method is based on electronic ledgers managed by trusted authorities, i. e. large financial institutions, that exchange electronic messages over a secure channel with other authorities to balance credits. The correctness of transfers can be considered to be close to perfect. Occasional failures may be recovered due to the lack of secrecy. The trust is based on legally enforceable contracts signed by authorities and customers.

The issue of secrecy in electronic transactions has been already recognised in the original work on blind signatures (Chaum, 1983). Around the year 1970, proposals were made to safeguard paper money from cloning by the integration of quantum bits (Wiesner, 1983).

If no trusted authority is assumed and consequently, no official transaction ledger, the correctness may be compromised due to contradictory transactions to different parties leading to double spendings. This problem has been systematically studied in the context of computer systems where it is known as the 'Byzantine Generals Problem' (Lamport, Shostak and Pease, 1982):

> The situation can be described as the siege of a city by a group of generals of the Byzantine army. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach an agreement.

The authors show that the problem can only be solved if either more than two thirds of the participants are honest, or if messages cannot be forged. Electronic transactions can be easily altered. Electronic signatures may indeed ensure the integrity of transactions in a system with a fixed number of participants. However, if participation is not restricted and no authority handles a registration, adversaries may easily overrule the honest majority using multiple fake identities. The scenario is a generalisation of the original problem and a practical solution is given with the distributed global append-only ledger *blockchain* that is the very core of the *Bitcoin* protocol which introduces a currency of the same name *bitcoin* (Nakamoto, 2008).

3.2.3.1 *Bitcoin*

The idea of Bitcoin has been coined in 2008.

Bitcoin is a *cryptocurrency*, that is a digital currency employing cryptography to secure transactions and to control the creation of additional units. No trusted authority is employed. The model of Bitcoin consists of peers of which the majority is assumed to be honest (cf. SPP) and disposes of the majority of computational resources. Peers communicate by message passing over a TCP/IP channel and discover other peers using DNS, gossiping among peers, and hard-coded seed peers.

A so-called *Bitcoin address* is the equivalent of a bank account or a wallet. It is identified by the hash of the public key of a peer's key pair. A transaction defines a set of input and a set of output addresses. It must be signed using the secret keys corresponding to the input addresses.

New transactions are announced publicly by gossiping to other peers and are grouped in sequential blocks if they are not in conflict with the recorded transactions in the blockchain. Every block needs to fulfil a mathematical criterion, that is a computational expensive puzzle based on slow hash functions. Valid blocks are then appended to the *blockchain*. The puzzle of a block depends on the preceding block in the blockchain and on the *Merkle root* all included transactions. The Merkle root is the result of subsequent pair-wise hashing along the *Merkle tree*, that is a binary tree with all transactions assigned to the leaf nodes (Merkle, 1988).

Although the puzzle is difficult to solve, its verification must be efficient. Bitcoin employs a *proof of work* based on the cryptographic hash function SHA-2. A cryptographic nonce in the block is successively altered until the SHA-2 hash of the block is numerically smaller than the current difficulty target. While it takes many attempts to find such a solution, for instance about $2 \cdot 10^{21}$ in May 2017, it takes only one attempt to verify the result. There can be forks in the blockchain. In such case, the longest blockchain is given priority. Manipulations of the order of blocks or its transactions change the linked blocks, respectively the Merkle root, and require the proof of work to be redone. Manipulated blockchain forks must be the longest in order to be accepted, which requires more computational resources than the honest majority. Consequently, the modification of a given block becomes exponentially more expensive while new blocks are appended. To verify that no double spending occurred, the blockchain is scanned for all previous transactions.

Note, that the puzzle is most efficiently solved on specialised hardware and only a minority of peers, the so-called *miners*, take on this role. However, it does not require special permission and every peer can become a miner at any time. Miners are incentivised by optional transaction fees offered to the miner by the transaction issuer and by a fixed amount of newly created bitcoins to be transferred to any Bitcoin address chosen by the miner, e. g. the miner's address. Of course, the miner must be first to successively append its block with completed puzzle to the blockchain to receive its reward. With no authority such as a central bank, this is the only mechanism to introduce new bitcoins into the system and this new influx is diminished gradually to have an upper bound of 21 million bitcoins (Perez-Marco, 2016).

3.2.3.2 *Online Voting based on Blockchains*

Online voting based on the blockchain is eagerly anticipated by some groups and is considered by the public sector (Boucher, 2016). The hopes are to employ the the blockchain as an immutable, append-only PBB that is powered by volunteers and voters and records cast ballots. So far, most online voting protocols such as Helios provide for a server acting as a PBB that must be trusted to provide a coherent view to all voters and constitutes a single point of failure.

Various prototypes and commercial solutions are or have been developed[2]. However, the actual protocols are either lacking essential properties or remain obscure. A common approach is based on so-called *coloured coins* that allow to associate digital assets to Bitcoin addresses. Consequently, asset ownership can be traded like the Bitcoin currency and (pseudonymous) ownership is publicly verifiable by following the previous asset transactions.

To construct an online voting protocol, every voter must initially own a coloured coin representing its eligibility. Those coins are then transferred to a destination that corresponds to e. g. a candidate. Transactions are publicly verifiable, so that aggregation and evaluation can be carried out by every voter. Though, the publicity of all transactions endangers the secrecy of the ballot. Coins can be linked back to the voter that has been identified during the registration phase to receive its coins. Therefore, protocols employ classical online voting secrecy mechanisms based on trusted authorities, anonymous channels and Mix-Nets. Different protocols have been proposed to allow for anonymous bitcoin transactions (Miers et al., 2013; Ibrahim, 2017) that may provide secrecy of the ballot in blockchain-based online voting.

Scientific results are very sparse. Zhao and Chan (2015) present a protocol for a fee-based binary vote to transfer all fees to the winner that is not based on colored coins, but carries out a SMC on an alternative blockchain of Ethereum (Wood, 2014) which comes with its own comprehensive programming language. Ethereum allows to enforce not only the correct execution of transactions, but also complex computations. This enforcement turns manipulation detection measures seen in various online voting protocols into prevention measures.

A similar protocol in which honest voters get their fees reimbursed has been reported to be tested and is briefly presented hereafter (McCorry, Shahandashti and Hao, 2017). The *Open Vote Network over Ethereum* provides for an authority that sends code defining the electorate, a number of time-outs, the obligatory registration fee and the voting question to Ethereum. Voters register to the voting and send therefore their deposit along with their public key and a zero-knowledge proof on their secret key to Ethereum. The authority launches the casting

2 https://github.com/domschiener/publicvotes, http://votem.com, http://www.bitcongress.org, http://followmyvote.com, http://cryptovoter.com, http://votosocial.github.io, http://blockchaintechcorp.com

phase which allows voters to send their encrypted vote and a zero-knowledge proof on the vote validity to Ethereum. Eventually, the authority notifies Ethereum, i. e. its miners, to compute the tally. The tallied outcome can be found by exhaustive research of a discrete logarithm. Only afterwards are deposits returned to the registered voters. The deposit is assumed to prevent malicious voters to drop out before the protocol converged. If so, no tally could be computed.

The authors claim to report of the first implementation of online voting on a blockchain. Their protocol provides maximum ballot secrecy, i. e. to reveal the vote of a given voter, all other voters would need to collude. The correctness is enforced by the peer network consensus of Ethereum. Unfortunately, the protocol is not scalable. The block size limits the voting to about 60 voters and the deposit may not impede all voters efficiently to drop out. Voters have to rely on the authority to launch the different phases. The protocol is *self-tallying*. Once all ballots are cast, every voter can compute the final tally. That means, the last voter to cast its vote, may compute the final tally before casting its vote. To address this fairness issue, the authors suggest to add an additional commit stage to prevent voters from changing their vote.

3.2.3.3 *Discussion*

Blockchain protocols such as Bitcoin, Ethereum and many others based on the same principles have a number of interesting properties. Foremost, they do not require a trusted authority. All transactions are public, so that the correctness of past transactions can be easily verified. Recent transactions may be challenged by a fork, but this becomes rapidly very unlikely. The secrecy may rely on anonymous communication channels and anonymous acquisition of coins, e. g. as a miner. In practice, peers rarely afford to become a miner and use instead an exchange who requires in many cases a prior identification. Once, a peer can be linked to a Bitcoin address, its bitcoins can be traced throughout their entire life cycle. If the secrecy relies on SMC as in (McCorry, Shahandashti and Hao, 2017), the robustness and scalability is significantly reduced, so that only boardroom votings with a limited number of participants can be carried out.

The research in the field of blockchains is very vivid with many propositions to improve secrecy, scalability, latency of transactions and solve the issue of wasted resources due to the proof of work (Yli-Huumo et al., 2016).

3.3 TAXONOMY

Cryptographic algorithms allow to limit the power of authorities in online voting protocols. Note, that there is a trade-off. Less trust in authorities must be assumed if authorities are less powerful, but with

PROTOCOL	SPECIALISATION	TOPOLOGY	DISTRIBUTED PHASES
Paper-based voting	none (flexible)	distributed	all
Helios	selected authorities	centralised	verification
SMC-based	none	distributed	all
DPol	none	structured, ring	all
SPP	random authorities	structured, tree	casting, aggregation
Blockchain-based	none (flexible)	distributed	all

Table 3.1: Quality of distribution of selected online voting protocols.

increasing protocol complexity, more technological trust is required. Using *Anonymous Voting* for example, voters do not have to present their ballot in clear, but have to trust the properties of the eligibility token.

Furthermore, online voting protocols may implement a separation of powers. For this, authorities may be assigned either equal roles like in *Random Perturbation* or *Homomorphic Encryption* where authorities carry out the same tasks, or different roles. In some *Anonymous Voting* protocols, one authority ensures the eligibility of voters while the other aggregates all ballots and produces the tally. If all authorities are equipotent and their number can match the number of voters allowing for an identification between authorities and voters, the protocol becomes essentially distributed. In between, there is room for different kinds of partially distributed protocols that we want to characterise as follows. Note, that the registration phase shall not be considered here. The emphasis of the analysis is on large-scale votings.

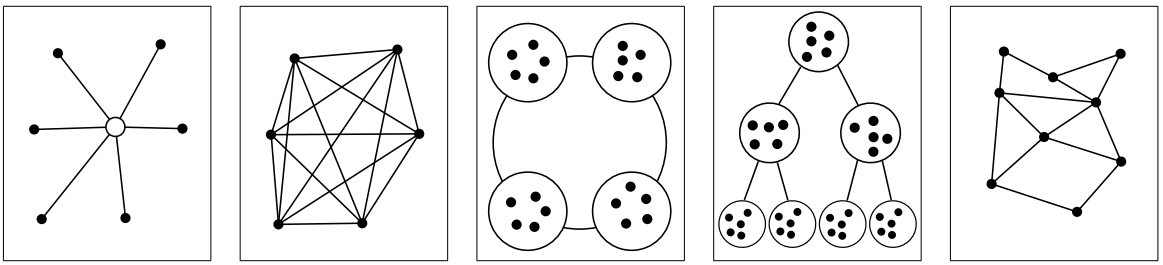
DEGREE OF SPECIALISATION ranging from equipotent voters with no specialisation to authorities with dedicated powers

TOPOLOGY ranging from centralised to distributed topologies, cf. Figure 3.2

PHASE Authorities can intervene not at all, only in few or in all phases (not distributed).

A protocol shall be called *fully distributed* if the topology is distributed and voters are equipotent during all phases but the registration. In contrast, online voting protocols with an aggregation phase that is not distributed to all voters, e. g. Helios, shall be called *classical online voting protocols*.

PAPER-BASED VOTING Paper-based voting is qualified as presented in Section 2.4. There are responsible voting officers, who can technically be exchanged as no special knowledge is required. The commu-



(a) Helios (b) SMC (c) DPol (d) SPP (e) blockchain-based

Figure 3.2: Topology of distributed voting protocols. (a) Helios has a centralised model. (c) DPol and (d) SPP are decentralised and have a higher degree of hierarchy than protocols based on (b) SMC or (e) blockchains due to partition of voters in clusters. This way, the number of exchanged messages can be reduced at the cost of less equipotent voters with less flexible roles. Unstructured SMC- and blockchain-based protocols use a distributed topology allowing for equipotent voters.

nication is public. If we accept that every voter can become spontaneously voting officer, the protocol is actually fully distributed during all phases.

HELIOS The Helios online voting protocol (cf. Section 3.1.4.1) based on *Homomorphic Encryption* relies on a central server (Figure 3.2a) to receive the ballots and publish the voting material for the public verification. Voters and authorities are distinct. The aggregation phase employs threshold cryptography to compute the final tally, but only very few individuals may serve as authorities. Eventually, only the verification can be carried out by all voters and is as such distributed.

SECURE MULTI-PARTY COMPUTATION (SMC) In protocols implementing SMC as described in Section 3.1.6, voters carry out the same tasks during all phases. In absence of a trusted authority, each voter exchanges messages with $O(n)$ other voters. SMC protocols are fully distributed.

DECENTRALISED ONLINE POLLING (DPOL) DPol and its extension EPol provide for equipotent voters. The aggregation is a joint effort in which all voters are involved. Every voter computes the final tally and then the voting result. To reduce the message complexity, voters are partitioned to groups that are alined on a ring. The structure is immutable during the voting. In consequence, the information on individual ballots is kept on a local level in cause of the introduction of intermediate tallies comparable to paper-based voting with many voting stations. On the downside, the voting is interrupted if the majority of voters of a

single, arbitrary cluster stop functioning. While the authors provide a special voter insertion routine to distribute malicious voters uniformly to all groups, there may be external factors affecting those voters, such as targeted DDoS attacks. The hierarchy has a negative influence on the protocol's robustness.

SCALABLE AND SECURE AGGREGATION (SPP) The online aggregation protocol SPP, that is based on SMC, achieves better scalability due to the introduction of a tree hierarchy and roles. Random peers in the cluster assigned to the tree root become occasionally authorities that employ threshold cryptography to generate the encryption key and decrypt the final tally. Roles cannot be swapped. The voting is interrupted if the threshold for the decryption cannot be reached for some reason. If peers assigned to clusters of internal nodes malfunction collectively, the intermediate tally of the respective subtree is not taken into account.

BLOCKCHAIN-BASED VOTING The blockchain offers a distributed PBB with similar properties to the publicity of paper-based voting. In both cases, the verifiability is achieved due to observation. The correctness is enforced by a consensus of the network. Voters are equipotent and all phases are distributed. The topology is inherited from bitcoin which uses gossiping to spread transactions (Figure 3.2e) in a network with no hierarchy. Note, that blockchain-based voting do currently not allow for large-scale elections.

3.4 SUMMARY

Classical online voting protocols represent the overwhelming majority among all published proposals to carry out online voting and increasingly achieve the essential security properties. The list of e2e verifiable and software-independent protocols is growing. Concepts are developed to provide not only computational ballot secrecy, but everlasting secrecy. At the same time, the potential for manipulations by voters or authorities is limited due to zero-knowledge proofs or separation of powers to a mutual distrusting set of authorities. A recent example (Locher and Haenni, 2016) employs anonymous channels, MixNets, homomorphic encryption and zero-knowledge proofs at the same time to ensure everlasting ballot secrecy assuming an adversary model of colluding authorities. Consequently, the involved complexity to realise such a voting protocol is high. This challenges not only programmers to provide flawless implementations, but also the voters that cannot convince themselves of the alleged protocol properties and are left to trust the assertions of experts. The mechanism to ensure correctness and secrecy remains opaque for voters with no expert knowledge. Nonetheless, the value of a transparent voting protocol has been stressed

repeatedly, i. a. by the German Federal Constitutional Court (cf. Section 2.5.2.1), and has been the motivation for the secret sharing scheme illustrating protocol *Threeballot* (cf. Section 3.1.5).

Moreover, in the light of hacks of political campaign servers in the advent of the 2016 US presidential elections, voices have been raised that the democratic machinery comprising the voting equipment and procedures to conduct elections should be considered as *critical infrastructure* (Shackelford et al., 2016). If elections are carried out on the internet, the infrastructure should withstand even sophisticated DDoS attacks directed to subvert the security and impede the pursuit. The *non-interruptibility* of ongoing elections is essential for the stability of democracies and is at stake if the voting protocol comprises a single point of failure. This is the case of classical online voting protocols and most protocols that rely on the availability and honesty of few distinct voters or officials. Albeit being distributed, also DPol and SPP are vulnerable for attacks due to their hierarchy that exposes a cluster of peers as indispensable. With no or insufficient flexibility of roles, unavailable peers may not be replaced to maintain the service. Fully distributed protocols offer greater resilience against attacks with individual targets even though the correctness may diminish in case of attacks.

The US Dept. of Homeland Security designated election infrastructure as Critical infrastructure only few months later in 2017.

The only two protocols that have been identified in Table 3.1 as fully distributed are paper-based voting and blockchain-based voting. The prospect of distributed online voting protocols motivates the aim of this work: to develop a digital counterpart to the well-studied and well-proven paper-based voting. This direction is reinforced by a comparison of the technological development of financial transactions that lead to Bitcoin, and likewise file sharing that lead to BitTorrent.

Perez-Marco (2016) emphasises that already before Bitcoin, a decentralized currency has existed for several thousand years. Metals such as gold and silver are widely acknowledged as precious due to their physical properties and scarcity and have been used for transactions already in around 2600 BC. The value of precious metals is inherent and does not depend on authorities. There is no trusted institution that controls the amount of metal in circulation or its transactions. A transaction, i. e. an exchange of the physical commodity, requires a face-to-face meeting or a mutually trusted bearer. Similar to paper-based voting, the mobility and flexibility of transactions based on physical commodities is very limited. In the 18th century AD, the economist John Law established the first French central bank and introduced *bank notes* during a financial crisis that caused a shortage of precious metals. Bank notes were initially backed by gold and silver reserves owned by the central bank. During the 20th century, this link was gradually abandoned in favour of *fiat money* which is backed only by trust in authorities, i. e. state-owned central banks. Consequently, bank accounts managed by third parties and transactions by message passing amongst cooperating third parties become feasible and greatly increased the mobility

The NGO WikiLeaks was almost entirely cut of from donations due to a controversial, unilateral stop of business relations by Visa Inc. and Mastercard Inc. and Paypal in 2010/2011. Only few months later, WikiLeaks started to accept Bitcoin donations.

and flexibility. In contrast, the assumed institutional trust has gradually been increased and motivated monetary concepts such as Bitcoin with no trusted authority.

Also file sharing was initially distributed. Software was distributed on physical media, e. g. floppy disk or CD, via various channels ranging from face-to-face meetings to magazine supplements. The server-centric FTP and HTTP protocols allowed then to increase the coverage to all internet users and to deliver updates in a timely manner. On the downside, the server and also the routing protocol DNS are most often subject to authorities. BitTorrent with its various extensions provides means to share files to a large audience with great resilience.

The WikiLeaks website was repeatedly unavailable due to unilaterally deleted DNS records and DDoS attacks. In 2010, Amazon.com removed WikiLeaks from their servers. WikiLeaks uses BitTorrent and Tor to improve the availability.

Despite the recent advances in online voting technology, the potential of distributed protocols for resilient and privacy preserving P2P aggregation in large-scale elections has only been started to be investigated.

4

BITBALLOT

Inspired by the design of both BitTorrent and Bitcoin, a novel distributed online voting protocol called *BitBallot* has been developed with the aim to avoid both trusted authorities and cryptography as much as possible. So far, the protocol has been disclosed only to a small audience (Frénot, Grumbach and Reimert, 2013, 2014). A corresponding patent has been approved recently (Reimert et al., 2016).

This chapter contributes to the dissemination of BitBallot and goes beyond the original work with regard to two aspects. First, the gradual introduction of important design decisions provides means to present the essential concepts of BitBallot to a non-expert audience. Second, issues are identified and protocol improvements and extensions are discussed.

4.1 DESIGN GOALS

Classical online voting protocols (cf. Section 3.3) rely on authorities to carry out the aggregation of ballots required to compute subsequently the final tally. To ensure a secure voting, either a globally trusted authority or technological trust in the employed cryptography needs to be assumed. In both cases, the ballots are gathered at the authorities which amounts to a centralisation of information and in consequence, of power. In the case of a breach of trust or faulty cryptography implementation, the secrecy of all ballots is at stake and the final tally is at risk of arbitrary manipulations. If the authorities are out of reach, the voting is interrupted. BitBallot shall provide for a distributed aggregation in which voters play an active role and information on ballots is not gathered remotely, but distributed uniformly to all voters. The amount of information that can be gathered by any voter shall be bounded. The correctness and secrecy shall be enforced by the voters themselves.

'The right to free speech means the government can't arrest you for what you say. It doesn't mean that anyone else has to listen to your bullshit […].'
—Cueball in xkcd.com/1357

Similarly to DPol (cf. Section 3.1.5.1), BitBallot shall not employ cryptography. Protocols without or with less cryptography are less complex. As such, those protocols are easier to implement and easier to understand with no expert knowledge. The more a voter has understood of a protocol, the less he or she is compelled to rely on trust.

Every voter shall have the power to initiate a voting. For this reason, BitBallot must not rely on the availability of all voters throughout the aggregation phase. The dynamics of the participation of voters is called *churn*. At last, BitBallot shall be sufficiently efficient to conduct large-scale elections.

With no constraints due to zero-knowledge proofs or homomorphic encryption, BitBallot may deal with a large variety of privacy preserving aggregations that exceed the primary use case of tallying ballots. BitBallot is a distributed aggregation middleware that allows to compute collectively an aggregation function over the inputs of network peers. Potential use cases comprise the acquisition of health care data and confidential information from distributed sensor networks and other networks related to the internet of things. In the following discussion, the terms *voter* and *ballots* are replaced by their more general counterparts *peers*, respectively *inputs*.

4.2 PRINCIPLE CONCEPTS

4.2.1 *Pull Principle*

There shall be no central authority to accumulate inputs. Instead, equipotent peers have to conduct the aggregation to compute the final tally.

At first, peers only know their own *input* of which they are the so-called *source*. Inputs must be exchanged among peers so that every peer can compute the final tally. For this input exchange, the distributed information dissemination strategies *pull gossiping* and *push gossiping* are considered. Both strategies imply message passing between peers, but are distinct in the initiator, which is the information recipient in case of pull gossiping, and the sender in case of push gossiping. Push gossiping has the disadvantage that peers have to track the state of foreign peers to avoid redundant communication. Moreover, attackers with superior communication resources may push more messages than honest peers and cause a disproportionate influence on other peers' final tally. In contrast, pull gossiping limits the influence of attackers if fake identities can be successfully suppressed. At their discretion, peers send remote procedure calls (RPCs) to arbitrary peers to pull a copy of an input. That means, attackers are limited in spreading malicious information by the number of pull RPCs they receive. As the recipient of RPCs are randomly chosen, attackers receive in average not more RPCs as honest peers. In consequence, the influence of attackers is proportionate in the case of pull gossiping.

'Trying to squash a rumor is like trying to unring a bell.'
—Shana Alexander

A very important innovation of BitBallot is its secrecy mechanism. Peers respond to pull RPCs with any input they know of. This can be either their own input or a foreign input previously pulled from another peer. An example is given in Figure 4.1. Generally, the receiver is unaware if the own or a foreign input has been provided. The links between foreign inputs and their sources are effectively broken.

This scheme can be compared to paper-based voting protocols, where ballots in the ballot box look the same and the voter that casted a given ballot cannot be revealed. Further, it is similar to secret sharing schemes. The source of the input is blurred due to the previous confidential

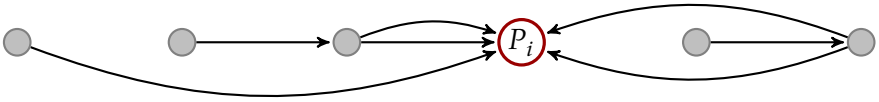


Figure 4.1: Exemplary flow of inputs to a peer P_i according to the pull principle of BitBallot. Peers (in gray) respond to pull RPCs from P_i. Note that P_i may receive different inputs from the same peer. Consequently, a response does not allow to link a peer to its input.

message exchanges. Secret sharing schemes rely on predefined valid shares. BitBallot instead does not require a decomposition in shares and is as such more flexible. In fact, all voting systems that are compatible with paper-based voting (cf. Section 2.3) are supported. Bitcoin uses also such gossiping techniques to propagate new transactions in the network. In consequence, the source of a transaction in terms of an IP address is difficult to trace back.

The inputs are assumed to have a unique feature that allows to recognise identical copies and remove them to prevent tallying the identical input more than once. For instance, an identifier chosen once by each peer for its own input may be used. The probability of identifiers chosen twice can be arbitrarily lowered by increasing the size of the set of valid identifiers.

The pull principle is fundamental and constitutes a major difference to most other voting or aggregation protocols that rely on ballot casting. Peers must offer their own, potentially encrypted input pro-actively to another peer or an authority. In contrast, BitBallot does not employ explicit ballot or input casting.

4.2.2 *Aggregation over a Tree*

BitBallot makes provisions to limit the knowledge of individual inputs that a single peer acquires during the aggregation. This is achieved by a particular information exchange protocol. The flow of information is constrained by a tree overlay network in which each peer P_i is assigned to a distinct leaf node with ID x_i. The pull principle is applied on each tree level. That means, an *intermediate tally* of each internal node is computed from the inputs or intermediate tallies of its child nodes, that have been acquired by pull gossiping. Intermediate tallies may be a concatenation of inputs, or e. g. for approval voting sums per candidate that do not allow any more to conclude on individual inputs. The intermediate tally of the root node is the *final tally* that comprises the inputs of all peers.

The idea of intermediate tallies is similar to DPol (Section 3.1.5.1) and SPP (Section 3.1.6.1), that use subsets aligned in a ring, respectively assigned to tree nodes. However, unlike in SPP, all peers are assigned

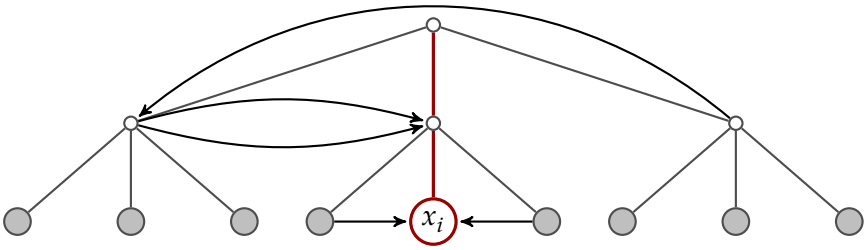


Figure 4.2: Exemplary flow of information to a peer P_i with leaf node x_i according to the pull principle of BitBallot on top of a tree overlay. Peers (in gray) respond to pull RPCs from P_i. Intermediate tree nodes represent any peer of the respective subtree.

to leaf nodes and no subset of peers is dominating or indispensable according to the idea of equipotent peers.

During the *aggregation phase*, peers cooperate in order to calculate jointly the final tally. All peers proceed at their own pace. As depicted in Figure 4.2, the peers of a subtree employ the pull principle to compute the intermediate tally of the subtree's root node. Note that while inputs have only one source, the intermediate tally is computed by several peers redundantly that become all sources of it. All peers start at the leaf node level of the tree and walk subsequently up the tree. First inputs, then intermediate tallies are exchanged. The aggregation is terminated when the final tally of the entire tree has been computed.

According to the pull principle, it is important for the secrecy that all inputs of leaf nodes, respectively, intermediate tallies of internal nodes with equal depth, cannot be linked to a particular node. For this reason, the arity k of tree nodes with equal depth shall be constant. Otherwise, a link may be established on the basis of the size of the tally. If the total number of peers n does not allow for a constant k, small deviations may be employed in tree levels preferentially close to the root where tallies comprise already many inputs and in consequence offer already a high uncertainty of the sources. The effect is especially for large n estimated as insignificant and this case not further discussed in the following.

The acquired information on inputs and intermediate tallies is considered for an aggregation over a tree with height h and with $n = k^h$ peers. Each peer learns a constant number of $k - 1$ foreign inputs from the set of its sibling leaf nodes. Peers cannot choose whose inputs they learn, because all peers are assigned randomly to leaf nodes by the authority. Moreover, the link between foreign inputs and their sources are blurred due to the pull principle. A guess of an input's source is only correct with a chance of 1 to $k - 1$ if distinct inputs are assumed. The provided secrecy can be improved further if the identity of the peers is protected using pseudonyms.

During the aggregation, each peer computes an intermediate tally of all its $h - 1$ ancestor nodes in the tree. Intermediate tallies of nodes at

depth d contain $O\left(k^{h-d}\right)$ many inputs. Thus, the chance to identify correctly the input of a source is for unique inputs at most 1 to $O\left(k^{-(h-d)}\right)$ it and decreases exponentially with every subsequent intermediate tally that is computed. Note that peers of non-sibling leaf nodes compute distinct intermediate tallies and acquire distinct information to compute the final tally. The information on inputs is bounded for each peer and distributed over the tree. In case of an intentional leak of a dishonest peer or an unintentional leak of a honest peer due to an attack, the information on inputs is limited to the partial knowledge of the respective peer, and thus the impact is limited, too.

4.2.3 *Aggregation Algebra*

Even though former work on BitBallot has not mentioned this explicitly, few assumptions on the aggregation algebra have been made that are all satisfied by FPTP voting and other common voting systems (cf. Section 2.3). To open the protocol for applications apart from voting, a generic algebra is introduced that operates on an application-specific data structure (Riemann and Grumbach, 2017b).

Aggregates are values to be aggregated, whether *initial aggregates*, constituting inputs from peers, or *intermediate aggregates* obtained during the computation, such as intermediate or final tallies[1]. The term *aggregate* for both inputs and tallies emphasises that inputs can be understood as well as a particular tally comprising exactly one input. The new nomenclature highlights the symmetry of the protocol with respect to initial and intermediate aggregates and is adopted consequently from now on.

Let denote $\mathbb{A}$ the set of aggregates. The aggregation operation, $\oplus$, combines two *child aggregates* to a *parent aggregate* in $\mathbb{A}$.

$$\oplus : \mathbb{A} \times \mathbb{A} \mapsto \mathbb{A} \tag{4.1}$$

By successive pair-wise application of $\oplus$, an arbitrary number of aggregates can be mapped to a single aggregate, so that an n-ary aggregation operator can be constructed. Consequently, the aggregate of n initial aggregates, one of every peer, can be computed and is called *root aggregate* a_R.

Note that due to the pull principle, all peers may retrieve the initial aggregates in an arbitrary order. To ensure that the aggregation result is invariant with respect to the order, $\oplus$ must be commutative. Moreover, applications such as voting require an aggregation result that is independent from the configuration of the tree, i. e. the partition of peers to leaf nodes that determines who contributes to which intermediate aggregates. Hence, $\oplus$ must be furthermore associative to allow for a unique aggregation result for any tree configuration.

1 Note that *final tallies* are in this work considered to be particular *intermediate aggregates*.

Considering the example of paper-based voting, an aggregate $a \in \mathbb{A}$ corresponds to a ballot box. $\oplus$ is the operation for joining ballot boxes of different voting stations to one common ballot box. An initial aggregate corresponds to a ballot box with only one cast ballot.

For illustrative purposes, the algebra for FPTP voting is developed. Peers, or here more precisely voters, choose one out of d options, that are modelled in the algebra with initial aggregate vectors $a_i = (e_1, \dots, e_d)$ in $\mathbb{A} = \mathbb{N}^d$. In voting applications, valid initial aggregates shall have a constant *aggregate weight* determined by the Manhatten norm $\|a_i\|$ in order to ensure the democracy and accuracy criterion (cf. Section 2.1.2). Without loss of generality, this work assumes a weight of $\sum_{x=1}^{d} |e_x| = 1$. The operation $\oplus$ is simply vector addition in $\mathbb{A}$. The root aggregate $a_R = (n_1, \dots, n_d)$ with $\sum_{x=1}^{d} n_x = n$ indicates how many peers n_x have chosen each option. The option x with the highest n_x, hence majority, corresponds to the vote outcome. The system can be easily extended to $\mathbb{A} = \mathbb{Q}_+^d$ to support vote splitting between two or more options.

More complex voting systems such as for instance the *Alternative Voting* and the *Single Transferable Voting* systems can easily be encoded. In both cases, voters have to rank options. Every ranking of $d!$ possible rankings in total can be interpreted as one option in the FPTP algebra presented above. The set of aggregates $\mathbb{A}$ consists of vectors in $\mathbb{A} = \mathbb{N}^{d!}$ and the operation is again vector addition. Note that alternative, more compact encodings can be defined to increase the efficiency, e. g. by indexing vector components and omitting in the aggregate those components with $n_x = 0$.

The aggregation algebra proves to be flexible, so that the protocol for confidential aggregations may also serve as a middleware for other applications than voting. Please refer to Section 2.6 for examples. Concrete examples are developed in Chapter 6 et seq.

4.3 PROTOCOL DESCRIPTION

The model of BitBallot consists of peers P_i, an authority A and at least one tracker T. The authority is the initiator of the aggregation and defines the set of eligible peers and the set of valid initial aggregates. Then, the authority has to inform all peers about the upcoming aggregation and carry out subsequently the registration of peers. The authority intervenes only during the preparation phase. To limit its power, the introduction of multiple distinct authorities has been investigated (Frénot, Grumbach and Reimert, 2013). However, it does not contribute to a separation of powers among peers and is therefore not in the focus of this presentation. The separation of powers among multiple authorities in the context of online voting has been studied previously (Fujioka, Okamoto and Ohta, 1993; Cramer, Gennaro and Schoenmakers, 1997).

Like most classical online voting protocols, e. g. Helios, BitBallot employs an adversary model for authorities in which only a subset of dishonest authorities deviate arbitrarily from the protocol, which does not meet a given threshold. With only one authority, the presentation below assumes the authority A to be honest. Peers are assumed to be honest, too. Models with adversarial peers are introduced in Section 4.6.

Peers and authorities communicate by message passing over pairwise bi-directional channels. The exchange of secret encryption keys with authorities requires a secure channel resistant to overhearing and tampering. In practice, this can be achieved with TLS/SSL over a TCP connection and a public key infrastructure (PKI) as deployed by some government-issued identity cards (Arora, 2008).

Even though BitBallot provides mechanisms to deal with unbalanced trees, the following discussion shall be limited for the simplicity of the presentation to *perfect full trees* whose leaf nodes are all at the same depth and whose node arity is constant. Therefore, n is assumed to be a power of the tree arity k.

4.3.1 *Preparation Phase*

After the aggregation has been announced to the peers, the authority A has to register every peer individually and confirm its eligibility which may happen only on prior request of each peer. The total number of peers n is known as soon as the registration has been closed. A proceeds to establish a perfect full tree of arity k and assigns each peer P_i to one distinct leaf node identified by some ID x_i. Further, A chooses a distinct symmetrical cipher key for each internal node of the tree. Each peer P_i is eventually provided:

- its unique leaf node ID x_i that allows to identify its ancestor nodes, e. g. `0.3.2.1` for child indexes separated by point delimiters;
- the cipher key for each ancestor node of x_i;
- a deadline for the preparation phase;
- at least one tracker T to discover other peers;
- the set of valid initial aggregates, among whose P_j chooses its own initial aggregate a_i, and which is used to verify a_j of other peers P_j.

As soon as all peers have received this information, the authority is not required any more. However, a tracker similar to those of BitTorrent is necessary to provide a mean to discover peers.

There is no casting phase. Every peer P_i prepares the aggregation by assigning locally its initial aggregate a_i to its leaf node x_i before the preparation phase deadline has expired. No communication is involved.

4.3.2 *Aggregation Phase*

The aggregation phase starts after the deadline of the preparation phase expired. Using the aggregation operator $\oplus$, every peer P_i with x_i computes the intermediate aggregate of each of its subtrees and assigns it to the root node of the respective subtrees. Remember that $\widehat{\mathbb{S}}(x_i, d)$ identifies the subtree whose root is at depth d and which contains leaf node x_i. To compute the aggregate of $\widehat{\mathbb{S}}(x_i, d-1)$, the own aggregate of $\widehat{\mathbb{S}}(x_i, d)$ and those of its sibling subtrees $\mathbb{S}(x_i, d)$ must be aggregated using the k-ary $\oplus$ operator. For this, P_i must request the $k-1$ sibling subtrees $\mathbb{S}(x_i, d)$ from other peers as depicted in Figure 4.2.

Similar to BitTorrent, peers can discover other peers using a dedicated tracker. The tracker provides on each request the contact informations (e. g. IP addresses) for a random subset of all peers. Note that no information on the peers' position in the tree is provided.

The aggregation is carried out in epochs, one tree level at a time. Epochs are loosely synchronized, because peers may have to wait for intermediate aggregates to be computed by other peers in order to continue. In each epoch for $d = h, \dots, 1$ with the tree height h, P_i computes the intermediate aggregate of the subsequent subtree $\widehat{\mathbb{S}}(x_i, d-1)$. Using tracker-provided information, P_i sends RPC messages to arbitrary other peers. The RPC specifies the subtree $\widehat{\mathbb{S}}(x_i, d-1)$ rooted at node N and may be responded with any aggregate assigned to child nodes of N. As only those peers P_j with x_j in $\widehat{\mathbb{S}}(x_i, d-1)$ are knowledgable to provide a response, many more requests than the $k-1$ sibling subtrees are generally required until all k child aggregates are gathered (cf. Figure 4.4a). Then, the intermediate aggregate of $\widehat{\mathbb{S}}(x_i, d-1)$ is computed using the k-ary $\oplus$ operator and the epoch is completed.

BitBallot provides for symmetrical encryption of the responses using cipher keys of subtrees. Encrypted aggregates can only be produced and decrypted by authorised peers in the possession of the cipher key. To allow for an exchange of aggregates of sibling nodes with depth d, the cipher key of the common parent node with depth $d-1$ is used for their encryption and decryption.

Further, every aggregate a of a subtree $\widehat{\mathbb{S}}(x, d)$ has assigned an ID h_a that is sent along with a. The union of (a, h_a) is called *aggregate container*. h_a is computed using a cryptographic hash function, e. g. SHA-3, and depends on the parent subtree $\widehat{\mathbb{S}}(x, d-1)$, the aggregate a and a list H_a of IDs of a's child aggregates.

$$h_a = h\left(\widehat{\mathbb{S}}(x, d-1), a, H_a\right)$$

The aggregate IDs provide a mean to distinguish between two aggregates a_1, a_2 of equal value of distinct subtrees $\widehat{\mathbb{S}}(x_i, d)$, $\widehat{\mathbb{S}}(x_j, d)$ that are both direct descendants of $\widehat{\mathbb{S}}(x, d-1)$ and have both to be aggregated. As the aggregate ID depends on the ID of all its child aggregates, a k-

ary *Merkle tree* is established that contributes to the integrity and verifiability of the aggregation (cf. Bitcoin in Section 3.2.3.1).

Initial aggregates a_i would have an empty list H_{a_i}, because aggregates of leaf nodes are not the result of an aggregation and have therefore no child aggregates. That means, if peers with sibling leaf nodes x_i and x_j choose identical initial aggregates $a_i = a_j$, their aggregate ID and aggregate container would be identical, too, and prevent any distinction. Hence, in the case of initial aggregates, a cryptographic nonce randomly chosen by each peer is used instead of a an empty list.

4.3.3 *Evaluation Phase*

Eventually, each peer P_i computes the aggregate a_R of $\widehat{\mathbb{S}}(x_i, 0)$, i. e. for the entire tree. If all peers are honest, an identical root aggregate container (a_R, h_{a_R}) is reproduced by all peers. Then, a_R can be used to derive the aggregation outcome. For illustration, consider the case of a FPTP voting among a set of candidates. Here, a_R expresses how many peers have voted for each candidate. The voting outcome is given by the candidate that has received the majority of all votes.

BitBallot does not allow for a verification of third parties. Peers compute their own root aggregate and verify implicitly that their own initial aggregate has been counted. Trust is assumed in the correctness of received aggregates.

4.4 ORIGINAL IMPLEMENTATION

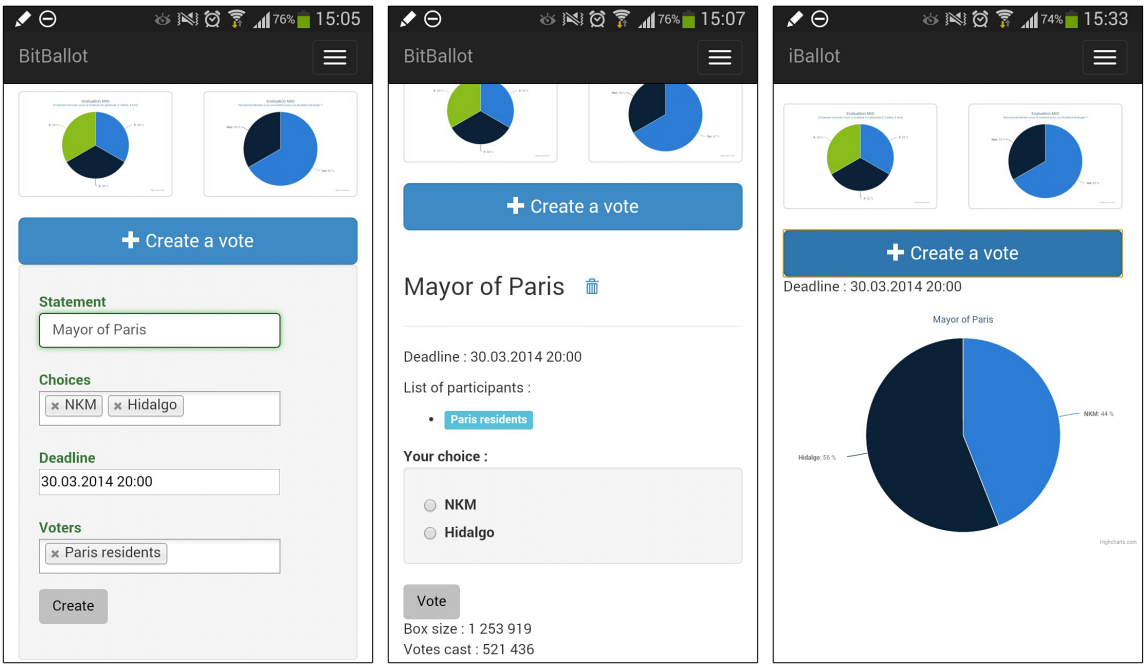
A demonstrator of BitBallot for FPTP voting has been implemented (Frénot, Grumbach and Reimert, 2013). It was realised on the basis of web technologies, in particular HTML5 and the JavaScript framework AngularJS[2]. The introduction of the WebRTC[3] API allowing for P2P applications in the browser has been anticipated already. However, the lack of wide browser support of WebRTC at that time required an alternative. Consequently, the WebSocket[4] API has been used as an intermediate solution that provides web pages a bi-directional communication channel to the remote server. That means messages between peers are relayed by a web server functioning as a proxy. The server components have been implemented in JavaScript, too. The language is event-oriented, which naturally fits the asynchronous character of the P2P protocol.

WebTorrent is a full-featured BitTorrent client that runs in the browser on the basis of WebRTC.

2 AngularJS (`https://angularjs.org`) is a JavaScript-based open-source front-end web application framework developed by Google.

3 *Web Real-Time Communication* (WebRTC, `https://webrtc.org`) is a W3C draft initially put forward by Google to allow for P2P connections among internet browsers using a simple JavaScript API.

4 WebSocket (`https://www.w3.org/TR/websockets/`) is a W3C candidate recommendation for a JavaScript API to allow two-way communication with a remote host.



(a) Preparation (b) Local Ballot Recording (c) Evaluation

Figure 4.3: Screenshot of the mobile application BitBallot. It has been implemented using HTML and JavaScript to run in the browser. In (a), a voter assuming the role of the authority specifies the voting question, a set of answers, a deadline and the electorate. Voters record their vote in (b). The computed voting outcome is presented in the evaluation screen in (c). Frénot, Grumbach and Reimert (2013).

All peers download the polster software from the BitBallot web server. Every peer can initiate a voting. For this, a peer P_A assumes simultaneously the role of both a voter and the authority and defines a voting question, a set of answers, a deadline and the electorate, cf. Figure 4.3a. Then, P_A registers the voting at the tracker. P_A generates the tree overlay, assigns all peers a leaf node ID and sends using the web server emails with the information required to vote to all peers. Peers register then at the tracker and record their ballot as shown in Figure 4.3b. Once the deadline of the preparation expired, the aggregation phase starts. As soon as the root aggregate has been computed, the voting outcome is presented in form of a chart as depicted in Figure 4.3c.

The implementation has been tested by students to grade university teachers. Further, automated tests have been carried out with up to 1000 simulated voters.

4.5 PROTOCOL PROPERTIES AND IDENTIFIED ISSUES

The authors of BitBallot emphasise that the contribution of BitBallot is foremost its pull gossiping mechanism and its unique distributed

aggregation algorithm that does not rely on the encryption of ballots to provide confidentiality. Then, essential voting protocol properties (cf. Section 2.1.2) could be generally enforced using the same cryptographic building blocks like classical online voting protocols. For instance, the aggregation can be encrypted using homomorphic cryptography to provide *fairness*. In the following, the properties of BitBallot are reviewed and analysed for their underlying assumptions.

4.5.1 *Security-Related Properties*

The *correctness* of the root aggregate is based on the tree structure in which each peer is assigned to one distinct leaf node. Although peers may receive aggregates more than once, redundant aggregates can be identified efficiently by their unique aggregate ID. The availability and responsiveness of all peers at the beginning of the aggregation is assumed. At higher tree levels, the fail out of an increasing number of peers is tolerated due to the redundancy of the computation. Further, peers are assumed to not manipulate aggregates and the authority is trusted to attribute every peer to a distinct leaf. *Integrity* must be assumed. Third parties cannot manipulate the aggregation without knowledge of the secret keys of intermediate nodes which ensures *eligibility*.

PROPERTY 1 (ACCURACY): The aggregation algorithm yields an accurate root aggregate a_R. Hashes and nonces are chosen sufficiently long, so that collisions can be neglected.

Proof (Sketch). Basis ($d = 1$). A trivial tree with depth $d = 1$ is considered. Once each peer P_i has its initial aggregate assigned to x_i and has contacted the tracker, each peer P_i requests repeatedly an aggregate from an other peer P_j. Given a non-zero probability to receive any aggregate, all k containers can be gathered. With no hash collisions and unique nonces, all aggregate IDs are unique and allow to distinguish distinct aggregate containers. Consequently, all peers find the identical exhaustive set of k aggregate containers. This set is complete, because of the bijection between initial aggregates and leaf nodes. With the identical set of aggregates and with the set operation $\oplus$, all peers compute the identical root aggregate a_R.
Inductive step ($d' = d + 1$). A non-trivial tree with depth $d' = d + 1$ can be disassembled in k separate trees of depth d, whose aggregation results are obtained accurately according to the basis case. The rest of the proof follows the same track as the proof for the basis case. □

The *confidentiality* of aggregates in BitBallot is based on the indistinguishability of aggregates of sibling child nodes, which is induced by the pull principle, the need of the cipher key to decrypt aggregates, and the bounded accumulation of initial and intermediate aggregates. It must be assumed that the cipher keys provided by the authority are

not initially compromised and that the peers do not compromise their own cipher keys.

If tracker-provided contact information such as an IP address and the leaf node ID x_i do not allow to reveal the identity of a peer P_i, there is no mean to link the initial aggregate a_i to its source P_i. The privacy of initial aggregates is trivially given.

In practice, the identification of a peer for a given contact information requires auxiliary information on the communication network, e. g. the internet, that institutions such as telecommunication providers possess, but which are not generally available. To lower the chance of identification, peers may further choose to exchange messages among peers only using anonymous communication channels provided by e. g. Tor (Dingledine, Mathewson and Syverson, 2004) or GNUnet (Wachs, 2015). The following analysis considers pair-wise bi-directional channels among peers as provided by e. g. TCP/IP and assumes that a link between the contact information and the corresponding peer can always be established.

According to the definitions in Section 2.1.2, BitBallot is not coercion-resistant, because a coercer-chosen nonce in the initial aggregate container can serve as a proof of compliance. BitBallot is only receipt-free to the extend that P_i can proof with an uncertainty of one to k to have provided one out of k initial aggregates to its first intermediate aggregate of $\widehat{\mathbb{S}}(x_i, h-1)$ which required to possess the corresponding cipher key. BitBallot alose does not ensure fairness, because all initial and intermediate aggregates of sibling nodes may be known to a peer P_i before P_i is requested for the first time to provide its initial aggregate a_i. Until then, P_i may change a_i. Though, the incorporation of homomorphic encryption in the aggregation algebra if possible, and the synchronisation of the aggregation using deadlines may allow or improve significantly the provided fairness.

PROPERTY 2 (CONFIDENTIALITY): The aggregation algorithm ensures confidentiality of a peer P_i's initial aggregate a_i.

Proof (Sketch). Due to the pull principle and the encrypted transmission of aggregates, only a limited random set of $k-1$ peers assigned to sibling nodes of a peer P_i may guess the initial aggregate a_i correctly with probability of 1 to $k-1$. Other peers receive information on a_i only in form of exponentially growing intermediate aggregates. □

Confidentiality does not assess if the acquired initial aggregates can be linked to their sources or not. BitBallot's pull principle allows peers to respond to RPC messages with own and foreign aggregates to blur its source. However, there is a moment when a peer has not yet acquired any other aggregates, so that its response on request contains its aggregate with certainty. Note that without knowledge on the prior communication of that peer, there is no certainty if this moment has been passed already or not. The probability $p(t)$ that a peer provides

its own aggregate as function of the number of its requests t is for one specific case evaluated in Figure 4.4b. One acquired foreign aggregate means already a significant drop to $p(t) = 50\,\%$. Though, the issue is foremost to find soon enough by chance a peer that belongs to a sibling node and can provide a foreign aggregate, which requires especially many attempts at the leaf node level and less and less with decreasing node depth.

The protocol incorporates software-independence and e2e verifiability (cf. Section 3.1) in the sense that peers can principally employ their own protocol implementation and carry out the aggregation at their own discretion. BitBallot provides only *individual verification* and peers are assumed to trust the correctness of aggregates computed by other peers. Note the similarity to paper-based voting, where the verification of tallies in voting offices run in parallel is also entrusted to voters in those offices. While the intermediate tally of such a voting office is already based on a consensus of multiple voters and ideally their local majority, this does not hold true for BitBallot. Hence, an issue arises if an adversary model with dishonest peers would be assumed. Then, peers may each verify a different intermediate or root aggregate due to manipulations by dishonest peers. The authors of BitBallot suggest that peers may use a PBB to publish their root aggregate which would allow for a detection of deviating root aggregates. However, the PBB requires again trust in the authority that it provides. Moreover, BitBallot does not provide a mean to find a consensus in this situation and without recovery mechanism, BitBallot provides only weak software-independence.

Small deviations of the root aggregate can be tolerated by the peers as long as the voting outcome is invariant. The margin depends here on the voting system and the set of initial aggregates.

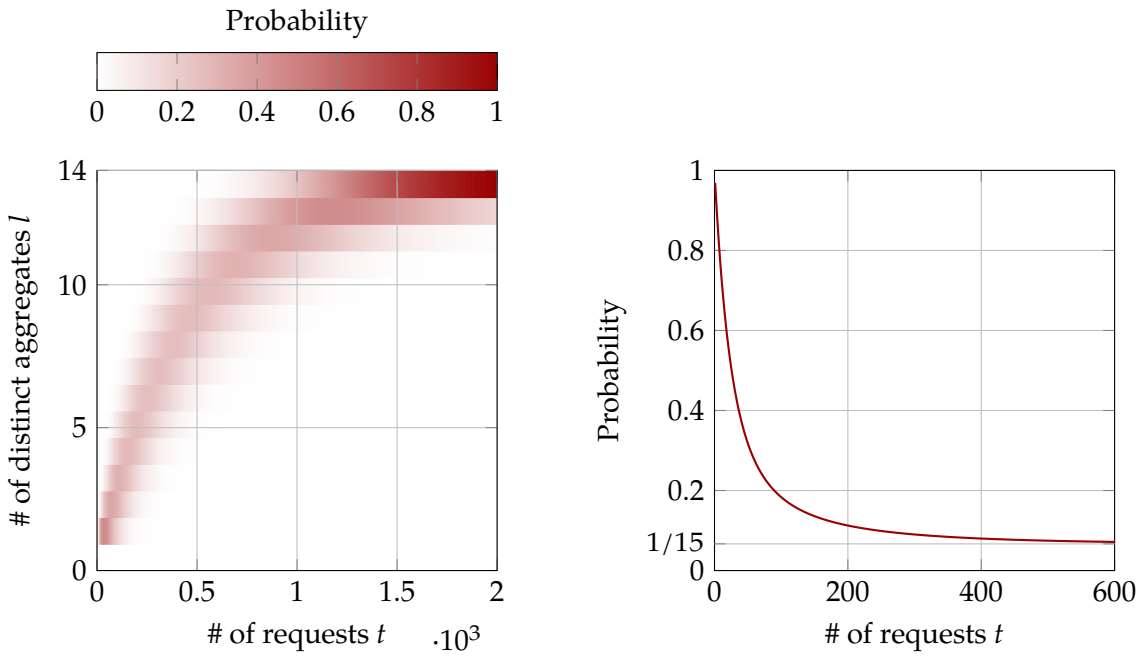
American scientists have found that the aggregation of paper-ballots by hand count exhibits an uncertainty of up to 2 % (Goggin et al., 2012).

As presented, BitBallot is not *robust*, because the aggregation stalls if one registered peer does not provide its initial aggregate. This issue and its potential solution based on time-outs have been already addressed (Frénot, Grumbach and Reimert, 2013). Though, time-outs require the assumption of a global clock as otherwise peers may compute different aggregates for identical nodes. An improved protocol is presented in Section 4.6.1.

4.5.2 *System-Wide Properties*

BitBallot does not impose any constraints on the locality of voting as long as the required communication channels can be established. The resulting *mobility* improves the *convenience* for the peers. Further, the employed aggregation algebra allows for various different voting systems providing therefore great *flexibility*.

To evaluate the *scalability*, the number of aggregation operations, the amount of RPC messages among peers, and the memory requirements



(a) Probability $p_{r,l}$ that a peer has l distinct foreign aggregates after t requests, cf. Expression (A.1).

(b) Probability $p(t)$ that a peer responds with its initial aggregate after t requests, cf. Expression (A.2).

Figure 4.4: Scalability and confidentiality of BitBallot. A tree with arity $k = 15$ and depth $d = 2$ is considered. P_i joins the aggregation when all 14 voters of sibling leaf nodes have already acquired their 14 aggregates. (a) shows the probability $p_{r,l}$ of having found l distinct aggregates of 14 in the set of r responses. The probability $p(t)$ that P_i responds with its aggregate after t requests is given in (b). A detailed description is given in Appendix A.

are determined. The total number of peers n of order $O\left(k^h\right)$\$, here $n = k^h$, is tied to the tree with height h and arity k. During the aggregation phase, each peer computes $h - 1$ intermediate aggregates with h of order $O\left(\log n\right)$. For each intermediate aggregate, up to a constant number of k aggregates must be stored, i. e. $O\left(k \log n\right)$ for all intermediate aggregates.

The number of RPC messages depends on two aspects. First, peers have to discover those peers that are sources of required foreign child aggregates. The tracker does not provide any information on the tree position of the peers. The average number of messages to find a first source of aggregates of nodes with depth d is n/k^d. The sum is developed for all node depths d from h to 1.

$$\sum_{d=1}^{h} \frac{n}{k^d} = n \cdot \underbrace{\sum_{d=0}^{h} \left(\frac{1}{k}\right)^d}_{< \frac{1}{1-1/k} = \text{const.}} - n < \frac{n}{k-1}$$

A second contribution to the number of messages arises out of the pull principle. Sources can respond with own or foreign aggregates which

leads to duplication. So more than the strict minimum of $k-1$ RPC messages may be required to complete the set of k aggregates to compute the intermediate aggregate of the common parent node. The analysis presented in Appendix A and the example provided in Figure 4.4a show that in the worst case the required number of messages exceeds k by far. For instance, assuming an arity of $k = 15$, about 1.250 messages are necessary to acquire all 14 foreign aggregates with a probability of about 50 %. For $k-1$ foreign aggregates and any number of responses, the number of distinct response sets is given by combination with repetitions and is dominated by exponential growth $O(\exp k)$ of which only one contains all foreign aggregates. In consequence, k must be small enough for an efficient aggregation, but not too small to hamper the discovery of sources and to diminish the confidentiality provided by the pull principle. For demonstrations, arity values of $k < 5$ have been used.

4.5.3 *Summary*

The confidentiality provided by BitBallot differs from classical online voting systems. It is not solely based on cryptography, but uses a particular sharing schemes (*pull principle*) to obfuscate the sources of aggregates. Encryption may optionally employed on top to further increase the confidentiality level.

However, other issues render BitBallot unsuitable for high-stake large-scale applications. The message complexity amounts to $O(n)$ per peer and $O(n^2)$ for the entire aggregation, which is impractical for large n. Further, the assumption of a honest authority and honest peers required for a correct aggregation outcome greatly limits the potential use cases and would allow for even simpler aggregation schemes with a star topology.

The protocol offers no protection against insider attacks. The authority may contribute initial aggregates on behalf of the peers, assign more than one peer to the same leaf node, or distribute wrong keys to individual peers. Dishonest peers may voluntarily disclose their keys with other nodes, manipulate arbitrarily intermediate aggregates and spread them on request. Peers are assumed to not fail out before their initial aggregate has been pulled by the network. Otherwise, the aggregation comes to a halt. Further, dishonest peers may covertly provide initial aggregates on behalf of registered, but absent peers. If more aggregates than child nodes of a given node are circulating, it is impossible to identify the illegitimate aggregates.

While the scaling and inside attacker are less important for small-scale applications such as board room voting with only few voters, the confidentiality properties are then inappropriate as chances are high that a voter provides its initial aggregate to a voter that it knows personally.

4.6 EXTENSIONS

In order to improve the properties of BitBallot as presented by its authors Frénot, Grumbach and Reimert (2013), different protocol extensions and variations have been explored with the goal of robust large-scale applications resilient against insider attacks by dishonest peers. The experiences led ultimately to the design of a novel distributed aggregation protocol presented in Chapter 5.

4.6.1 *Absent Peers*

The pull principle relies on the presence and responsiveness of the peers. The *convergence* is at stake if repetitive pull requests to random peers assigned to sibling nodes do not allow to obtain eventually the entire set of k aggregates that is required to compute the subsequent parent aggregate. This is especially an issue at the beginning of the aggregation when the lack of initial aggregate may block the entire aggregation.

Thus, BitBallot is extended by provisions to ensure the convergence of the protocol in the case of intentional or unintentional absence of peers. Unintentional absence can be caused by shortcomings of e. g. the underlying communication channel or a fail out of the peer. Peers are either present or absent. Absent peers may leave at every moment after the preparation phase. It is assumed that every subtree with child nodes comprises at least one present peer.

The authors of BitBallot addressed the issue of peers that do not provide their initial aggregates before a given deadline t_d. They suggest that peers employ an empty placeholder aggregate for every aggregate missing after t_d. Though, with no global time, peers may act differently upon foreign aggregates that they receive close to t_d due to shifts of their local clock. Further, it is unclear how all peers in the same subtree can get a missing aggregate still before t_d that only one of them or a small subset acquired just before t_d. In consequence, different aggregates are computed for the same node N and subsequently for its ancestor nodes. Thefore, a new approach is developed that allows peers to converge to the identical aggregate of N.

The aggregate container consisting of aggregate a and its ID h_a is extended by an *initial aggregate counter* c_a to (a, h_a, c_a). The counter c_a denotes the number of included initial aggregates in a. $c_{a_i} = 1$ for initial aggregates a_i. For computed aggregates a of internal nodes N, $c_a = \sum_\alpha c_\alpha$ with α a child aggregate of a. The counter c_a is a measure for the completeness of a. If c_a equals the number of descendant leaf nodes of N, a is called *complete*, otherwise *incomplete*. An initial aggregate is always complete per construction.

While there is only one distinct complete aggregate for every node, there can be different incomplete aggregates for internal nodes due to

different subsets of included initial aggregates. To ensure the correctness of the aggregation, not more than one aggregate of every child node of N must be used to compute the intermediate aggregate for N. The container selection based alone on unique aggregate IDs is inappropriate as incomplete aggregates of the same child node have diverging IDs. Therefore, an improved selection criterion is devised.

All acquired aggregate containers of any child node of N are grouped to so-called *container clusters* by their assigned child node. Due to the pull principle, the actual child node cannot be directly deduced for a given aggregate container as only the parent node is given. Hence, a heuristic clustering method shall be used. For this, aggregate containers are again extended by the set H_a of its child aggregate IDs to (a, h_a, c_a, H_a). The authors of BitBallot introduced also such a set of child aggregate IDs, but for the purpose of verification and not to support the convergence.

Child aggregate containers a_1, a_2 are supposed to origin from the same child node N_c if they comprise at least one common grand child aggregate. Neglecting hashing collisions, the pair-wise intersection of $H_{a_1} \cap H_{a_2}$ must be non-zero. Clusters might be merged later on as new containers are acquired with child aggregate IDs that have been initially found only in distinct clusters.

Furthermore, each peer P_i has for all ancestor nodes N of x_i a so-called *own cluster* which contains among others all child aggregate containers whose source is P_i itself.

Generally, the container with the highest counter c_a of each cluster is *confirmed* as the most accurate. An additional condition applies for each P_i's own container. Here, only those containers are considered for confirmation whose source is also P_i, i. e. that have been computed and thus verified by P_i.

If other containers in the own cluster are more complete than the confirmed container, P_i may have missed out in previous epochs child aggregates. It goes recursively back to the previous epoch and starts requests again to find new or more complete child aggregates. If the confirmation status changed, all already computed aggregates of ancestor nodes must be updated accordingly.

There is at most one confirmed container per cluster. The subset of maximal k most complete confirmed containers, which must contain the confirmed container from the own cluster, is used to reply to aggregate requests and compute the parent aggregate. In the case of two or more distinct incomplete containers that have the same highest counter, no container shall be confirmed, but requests continue to find and confirm a more complete container.

This extension allows the protocol to converge even in the case of absent peers or missing initial aggregates. It increases the protocol robustness, but harms the efficiency as the message volume increases due to updates. The accuracy suffers because not all initial aggregates of all

present peers may be included in the root aggregate after the protocol terminates when the deadline for the root aggregation elapsed. The upper-bound duration until convergence has not been examined and is an open research question. However, with unbounded time, the root aggregate is expected to be *eventually accurate*, i. e. the root aggregate converges monotonically to the accurate aggregate.

PROPERTY 3 (ACCURACY): The protocol extension for absent peers delivers a root aggregate a_R that is eventually accurate.

Proof (Sketch). Given Property 1, it must be further shown that (a) more complete aggregates replace always less complete aggregates and (b) incorrect aggregates that include initial aggregates more than once can be corrected. (a) is fulfilled in the case that the aggregates belong to the same cluster due to the confirmation algorithm. Incorrect aggregates emerge when different incomplete aggregates of the same node persist simultaneously over more than one level and are not clustered, because of entirely distinct H_a. By assumption, one peer is present in every subtree. So at least k clusters with complete or incomplete computed aggregates can be found. If two or more clusters for the same child node emerge, aggregates of valid clusters are omitted, which can be detected by effected peers. Those can eventually compute a parent aggregate that is more complete. Hence, (b) is satisfied. A formal proof may seek to introduce a global monotonic function to describe the global convergence behaviour. □

4.6.2 *Dishonest Peers*

The adversary model of BitBallot assumes honest peers. A different model is given with *honest but curious* peers, hereafter also called *curious peers*, who do not deviate from the protocol, but seek to deduce more information from the data at hand. While a curious peer alone does not learn anything more than any other honest peer, the situation is different for collusions of curious peers. If by the end of the aggregation, curious peers collude, the confidentiality may be harmed due to the deduction of information that exceeds the mere sum of the colluding peers' information. Consider the case of $k - 1$ colluding curious peers associated to sibling leaf nodes. Here, the colluding peers may substract their individual child aggregates from their parent aggregate in order to reconstruct the child aggregate of the honest k-th peeer. The situation can occur anywhere in the tree, but with aggregates that contain already an large number of initial aggregates, the chance to link contained initial aggregates to their sources becomes negligible.

A more intrusive adversary model provides for Byzantine or *dishonest peers* that deviate arbitrarily from the protocol. It is assumed that dishonest peers are globally a minority and do not collude if not stated explicitely. By assumption, at least one peer is honest in every subtree

child nodes. Dishonest peers may send requests to acquire aggregate containers for any node N, but cannot decrypt the request response if they are not in possession of the secret cipher key. BitBallot employs cipher keys for each node N to be shared with those peers that are assigned to descendant leaf nodes of N. Aggregates of node N are consequently encrypted using the key of the parent node of N and can only be decrypted by peers meant to compute the aggregate of N. Hence, BitBallot ensures the confidentiality of aggregates in the presence of dishonest peers.

Further, dishonest peers can manipulate aggregate containers and provide inconsistent responses on successive requests in order to introduce more than one aggregate ID and have its aggregate counted more than once. With more than k different child aggregates for a k-ary node N, peers gather a random subset in which aggregates of honest peers may be replaced by forged ones. Hence, honest peers must be provided a mean to detect manipulations.

In the following, different techniques with verifiable authenticated commitments on aggregate containers are considered to detect manipulated aggregates and allow for provable objections. For this, an asymmetrical cryptographic signature scheme is used. The following notation is employed:

(pk_i, sk_i)	Public-private key pair of peer P_i
$\sigma_i(m)$	Peer P_i's signature scheme using (pk_i, sk_i)
$\sigma_A(m)$	Authority's signature scheme
$\chi(m, r)$	Blinding technique for message m and random number r
$\delta(s, r)$	Retrieving technique of blind signature

4.6.2.1 *Signatures for Initial Aggregates*

The security model of non-colluding dishonest peers is assumed. To detect multiple initial aggregates provided by the same dishonest peers, digital signatures are employed. Similar to the concept of anonymous voting (cf. Section 3.1.2), blind signatures (Swanson and Stinson, 2011) are used to provide secrecy during the peer registration. The technique $\chi(m, r)$ allows to blind messages m with a blinding factor r and technique $\delta(s, r)$ to unblind s with the identical blinding factor r.

The registration of peers is extended by two additional steps. First, the authority A and each peer P_i generate initially on their own discretion a public-private key pair (pk_A, sk_A), respectively (pk_i, sk_i), to sign messages m with the private key using the scheme $\sigma_A(m)$, respectively $\sigma_i(m)$. The public key of the authority pk_A is provided to all peers. Second, the peers require a signature of the authority on their public key. The authority is already entrusted to verify the eligibility and is therefore further assumed to provide to each peer once such a signa-

ture to testify its eligibility. In consequence, each peer P_i is bound to one public key pk_i which can be interpreted as a pseudonym of P_i. The required request response cycle between P_i and A proceeds as follows:

1. P_i chooses randomly a blinding factor r_i, computes $b_i = \chi(pk_i, r_i)$ and sends it to A.
2. A receives b_i and responds to P_i with $s_i = \sigma_A(b_i)$ if it has not yet previously signed a key for P_i. Hence, A is bookkeeping its responses.
3. P_i unblinds s_i with r_i to get its *authorisation token* $t_i = \delta(s_i, r_i)$.

Peers are required to provide with any of their signatures their public key pk_i and their authorisation token t_i, so that both the signature and the key can be verified.

All aggregate containers of a are extended by the list H_a of IDs of a's child aggregates as in the previous extension: (a, h_a, H_a). Moreover, initial aggregate containers of a_i with aggregate ID h_{a_i} are provided a signature $s_{a_i} = \sigma_i(h_{a_i})$ from their source P_i. For the aggregation of initial aggregates, peers consider only one aggregate per public key and only if the public key is authorised, i. e. a valid t_i is provided. If more than one initial aggregate has been signed by a peer, none of its initial aggregates should be taken into account. Two signatures of distinct initial aggregate containers from the same pk_i, hence the identical peer P_i constitute a proof of a protocol deviation by P_i that can be used to convince other peers or third parties. Note that pk_i cannot be linked to P_i due to the blind signatures scheme, so that P_i is deprived of its eligibility to contribute to the aggregation, but cannot be prosecuted outside the scope of the aggregation algorithm.

The introduced signature scheme for initial aggregate containers incentivises dishonest peers to sign and submit only one initial aggregate in order not to risk to be excluded entirely from the aggregation. Dishonest peers are impeded with great probability to tamper with the first intermediate aggregate. If there is at least one honest peer in every subtree, honest peers could in theory reveal all their acquired initial aggregates to all other peers in order to rule out any manipulations by dishonest peers. Unfortunately, this would undermine the confidentiality and have an enormous efficiency drawback and is therefore impossible. Eventually, dishonest peers may not sufficiently tamper with initial aggregates, but with the subsequent intermediate aggregates.

In the case of colluding dishonest peers that are not assigned to sibling leaf nodes, private keys may be exchanged so that the limitation of one initial aggregate per peer in the computation of the first intermediate aggregate can be circumvented. In this adversary model, the presented signature scheme is not effective.

4.6.2.2 *Signatures for all Aggregates*

So far, dishonest peers can successfully tamper with intermediate aggregates to impede the emergence of a unique root aggregate. A straightforward approach to improve the accuracy is to build upon all previous extensions and require sources to sign not only their initial aggregate containers, but also their intermediate containers throughout the aggregation. This may open doors to link the aggregates signed with the key pk_i to their source P_i by a statistical analysis of key frequencies in responses. The uncertainty of the source due to the pull principle is undermined. To oppose such analyses, sources gather the signatures of other sources of their aggregates, and also of foreign aggregates. To respond to a request, peers choose first a random aggregate a and afterwards one random signature of a. That means, the aggregate container selection mechanism of the pull principle described in Section 4.2.1 is reused on the level of signatures.

Aggregate containers are clustered by child node using child aggregate IDs, as presented in Section 4.6.1, and public keys associated to aggregate signatures. The initial aggregate counter c is alone not a suited criterion any more to confirm a container of a cluster, because it can be subject to manipulations, too. Instead, aggregate containers are confirmed based on the number of signatures from distinct peers. Sources sign their aggregates as presented in Section 4.6.2.1. As shown in Figure 4.4a, the number of requests to find all k aggregates is much larger than k, so that multiple signatures of the identical aggregate from distinct sources can be accumulated with no additional requests.

The subset of maximal k most signed and most complete confirmed containers, which must contain the confirmed container from the own cluster, is used to respond to aggregate requests and compute the parent aggregate. In fact, a few other aggregates from the own cluster, that have less signatures, but are more complete than the confirmed aggregate, must also be used to respond to aggregate requests in order to promote them by providing potentially a new signature.

Similar to the correction scheme of Section 4.6.1, peers are recomputing their intermediate aggregates if they find by comparison, that another container in the own cluster with equal or higher value c_a has received more signatures.

PROPERTY 4 (ACCURACY): The protocol extension for non-colluding dishonest peers delivers a root aggregate that is eventually correct. It allows honest peers that confirmed erroneous aggregates to detect posteriori alternative parent aggregates with more signatures in their cluster which triggers a recursive correction.

Proof (Sketch). Manipulated aggregates a^* of dishonest peers have in most cases only one source, because dishonest peers are assumed to not collude and two identical, but independent manipulations are unlikely. There cannot be more signatures of a^* than sources. In consequence,

correct aggregates have in average more signatures and are more likely to be confirmed. If gossiping of signatures takes too long, honest peers may temporarily confirm a manipulated foreign aggregate and compute *erroneous aggregates*. Erroneous aggregates can have more than one source. However, pull gossiping puts bounds on the spreading rate of all aggregates and every manipulation or error decreases the potential number of signatures of unaffected peers. Eventually, the aggregate with the most sources is expected to prevail, which is the correct aggregate under the assumption of a honest majority. The rest follows as described in Property 3. □

The accuracy has been experimentally confirmed using a simulation presented in Section 4.7. Chapter 5 provides further concepts to improve the resilience against dishonest peers that may also be integrated in BitBallot.

PROPERTY 5 (INDIVIDUAL VERIFIABILITY): The protocol extension for dishonest peers ensures individual verifiability and allows to proof objections of the root aggregate while maintaining an uncertainty of the own initial aggregate of at least 1 out of k.

Proof (Sketch). Consider a honest peer P_i with root aggregate $a_{R,i}$ that includes its initial aggregate a_i. If another peer P_j proposes a diverging root aggregate $a_{R,j}$, it can be objected by P_i by revealing publicly all its aggregates including their signatures that led to $a_{R,i}$. To prevent that a_i may be linked to P_i by a heuristic frequency analysis of signatories, P_i may not reveal all of its signatures.

The many signatures of distinct peers on intermediate aggregates suggest that P_i participated in each epoch.

P_j may contest the objection by revealing its aggregates and the objection shall be rejected if P_j can provide as many signatures as P_i for the smallest diverging intermediate aggregate of $a_{R,i}$ and $a_{R,j}$.

The objection relies on a comparison of the number of signatures that relates to a consensus among those peers who signed. As the majority of peers is honest, aggregate containers of P_i have with high probability more signatures than their manipulated or erroneous counterparts of P_j. Peers other than P_i or P_j may provide further signatures if available for or against the conflicting aggregate for a more significant decision. □

4.6.2.3 *Restricting Signature Validity*

The adversary model of colluding dishonest peers is now assumed. The protocol extension in Section 4.6.2.2 is not sufficient to ensure the correctness of the root aggregate in this model. Dishonest peers P_i may exchange their keys pk_i after the preparation phase, so that one dishonest peer can issue many distinct signatures on a manipulated aggregate

container to overrule the aggregate of the honest peers with a majority of signatures.

Hence, a new extension to BitBallot is developed. The generation of the authorisation token t_i from Section 4.6.2.1 is modified in such a manner that t_i is linked to a leaf node x_i. For this, a partially blinded signature technique $b(m;x)$ is employed that only blinds a message m, but allows the authority A to verify some value x (Abe and Okamoto, 2000). Here, the value is the ID x_i^p of the parent node of each P_i's leaf node x_i.

P_i chooses randomly a blinding factor r_i and computes $b_i = b(pk_i, x_i^p)$. Then, b_i is sent to A which verifies that x_i^p is the parent of x_i and as before, no other signature has been issued for P_i. After unblinding, P_i has a signature t_i of A on both (pk_i, x_i^p). Consequently, both (pk_i, x_i^p) must be provided so that t_i can be verified.

Henceforth, a signature of an aggregate container is only valid if the container's parent is x_i^p or its ancestor node. That means that aggregate IDs must be formed in a way that allows to deduce the IDs of all ancestor nodes. This is given with ordinals in the basis of tree arity k that are in size equal to the node depth, e. g. `3.2.0` for a node with $d = 3$ in a tree with $k = 4$ and dot delimiter between all ordinals.

As a result, colluding dishonest peers cannot any longer use arbitrary keys to sign their intermediate aggregates and intermediate aggregates of honest peers cannot be overruled any more if the dishonest peers have no majority in the local subtree.

While this extension improves the correctness in face of colluding dishonest peers, it undermines the secrecy provided by the pull principle. Due to x_i^p, child aggregates can be linked with certainty to one subtree. The secrecy provided by the pull principle is only effective for the exchange of leaf node aggregate containers.

The secrecy could be re-established if the authority would provide every peer not with a single authority token t_i, but with as many tokens as the tree height h. Of course, those tokens would have to be bound to a given tree depth. Their introduction lead to an increased computational complexity for the authority from $O(n)$ to $O(nh) = O(n \log n)$.

This extension has not further been investigated or implemented, because both the introduction of keys per peer and per node or the renunciation of the pull principle are not in alignment with the aim to provide a confidential aggregation protocol with less complexity for a better comprehension.

4.6.2.4 *Legitimate Requests*

Dishonest peer may send illegitimate pull requests for any tree node N that is not an ancestor of their leaf if they know the ID of N required for the request. Request responders cannot deduce the legitimacy from the request and respond in any case. Responses are encrypted, so that

the confidentiality is not harmed, however, an unnecessary computational and communication load is placed on the responder. BitBallot is further extended, so that the legitimacy of requests can easily be assessed.

Unlike as suggested in Section 4.6.2.3, node IDs shall not follow an ordered scheme such as `0.3.2` that allows to guess IDs easily. Instead, a scheme based on secrets is employed. One option is to have the authority A assign a random hash to each node and construct the node ID by a concatenation of all ancestor nodes starting from the root, e. g. `D59.FAE.67B`.

Still, a dishonest peer may learn about foreign IDs by the requests it receives. This can be encountered by using only hashed node IDs $h(ID_N, C_{ij})$ tight to the request receiver P_j by the channel identifier C_{ij}, that is assumed to be known by the request sender P_i and P_j and uncontrollable by P_i. In the case of a P2P network based on the TCP protocol, C_{ij} could be the IP address of the receiver P_j. As P_j knows C_{ij} and the node IDs of its leaf node x_i and consequently all its ancestor nodes, it can generate in advance a table of all hashed node IDs. With this table, P_j can relate a received request specifying a hashed node ID to an actual ancestor node. If the hashed node ID is unknown, P_j does not learn the foreign node ID. The link to C_{ij} prevents that hashed foreign node IDs may be reused by dishonest peers for illegitimate requests.

Colluding dishonest peers share by definition all their knowledge and this includes their leaf node IDs, which can in this scenario be used to send requests on behalf of colluding dishonest peers, or already acquired aggregates.

4.7 IMPLEMENTATION OF EXTENSIONS

An extended BitBallot protocol has been implemented on the basis of the Java software library SIMGRID dedicated to 'Versatile Simulations of Distributed Systems' (Casanova et al., 2014). The focus of the original implementation has been the feasibility to use HTML5 standards only and to provide a usable demonstrator for first tests outside the lab. Even though that implementation was prepared to run the same code in a simulation setup, different aspects such as an extensive documentation, a vivid community and ongoing updates lead to reconsider the strategy and use the dedicated and well-established simulation software SIMGRID instead.

The simulation software denoted *simgrid-bitballot* implements the extension for absent peers from Section 4.6.1 and for dishonest peers from Section 4.6.2.2. It allows for a tree with a configurable tree arity k. The arity on leaf level k_l to determine the number of sibling leaf nodes can be controlled separately. Depending on the number of peers n, the height of the tree is automatically determined. The simulation takes into account that a tree might not be full and seeks to balance subtrees of higher depths first. The aggregates are all empty and no aggregation

operation $\oplus$ was defined for the sake of simplicity. The correctness of the result is evaluated on the basis of the Merkle tree. All honest peers should produce a root aggregate container with an identical aggregate ID. Further, the signing scheme is assumed to be robust and is likewise not implemented for efficiency reasons. Unique peer IDs are used instead of public keys pk_i and authorisation tokens t_i, which are in the simulation not manipulated by dishonest peers.

Indeed, signing implemented in JavaScript works in the browser, but it also slows down the simulation dramatically.

The modelling of dishonest peers proved to be difficult as ideally all possible manipulations must be taken into account. Instead, the implementation is limited to *fuzzing*. That means dishonest peers respond to requests with partially random aggregate containers. Those containers whose manipulation can be detected with certainty, have not been taken into account. This is for example the case for aggregates with c_a exceeding the number of child aggregate IDs in H_a, $c_a < 1$, aggregate IDs h_a that cannot be reproduced, and so on. Further, also initial aggregates are excluded from fuzzing, because the signatures introduced in Section 4.6.2.1 allow to detect manipulations with high probability and spread the proof of protocol deviation as part of the pull gossiping. Eventually, the fuzzing impacts the initial aggregate counter c_a and the set of child aggregate IDs H_a that both influence h_a. Absent peers are modelled to be silent (fail-stop) right after the preparation phase. Dishonest peers are always present.

The simulation is deterministic. The Java pseudorandom number generator can be seeded differently, so that the choice of requested peers and aggregate responses can be varied. Simulations have been carried out with a few hundreds of peers. The code of the implementation has been published under the license LGPL v2.1 and is available at `https://gitlab.inria.fr/riemann/simgrid-bitballot`.

The simulation allows to evaluate the performance of the protocol in a controlled environment. Aggregations have been carried out for $n = 135$ peers comprising varying ratios of absent and dishonest peers. The results are provided in Table 4.1. A main concern has been the protocol convergence. The convergence behaviour can be assessed by the number of distinct root aggregates $\#\, a_R$ among the set of all honest peers. When the aggregation converges, many, if not all, honest peers compute an identical a_R, i. e. $\#\, a_R$ is small. Large values of $\#\, a_R$ indicate a diverging aggregation.

While the exhibited behaviour is not yet fully understood, the data confirms that a convergence is possible even with close to 50 % dishonest peers employing fuzzing among all present peers. The precise ratio of dishonest to present peers that still allow for a convergence depends on the ratio of present to all registered peers. One observes that with a smaller ratio of present peers, the aggregation becomes more susceptible to manipulations by dishonest peers. A reason may be that dishonest peers succeed in impersonating absent peers that do not provide conflicting data to activate the correction. With no dishonest peers, the

The voter turnout in the 1st round of the French legislative election 2017 has been 48.71 %.

n_{present}	$n_{\text{dishonest}}$		# a_R	$f_{\max}$ of a_R		γ_{a_R} (REL. c_{a_R})	$\gamma^*_{a_R}$ (REL. $c^*_{a_R}$)	# REQ.
38	0	(0 %)	1	38	(100 %)	1	0	220(10)
38	7	(18 %)	23	3	(10 %)	3.51(8)	3.4(1)	330(130)
54	0	(0 %)	1	54	(100 %)	1	0	208(5)
54	10	(19 %)	28	4	(9 %)	2.49(1)	2.39(5)	220(130)
54	21	(39 %)	30	2	(6 %)	2.50(1)	2.44(1)	240(120)
75	0	(0 %)	1	75	(100 %)	1	1	208(5)
75	15	(20 %)	2	59	(98 %)	1.0(1)	0.0(2)	220(20)
75	30	(40 %)	37	2	(4 %)	1.80(1)	1.71(5)	280(150)
105	0	(0 %)	1	105	(100 %)	1	0	205(3)
105	21	(20 %)	1	84	(100 %)	1	0	213(8)
105	42	(40 %)	1	63	(100 %)		0	220(10)
105	50	(48 %)	3	53	(96 %)	1.01(5)	0.1(2)	230(20)
135	0	(0 %)	1	135	(100 %)	1	0	204(3)
135	27	(20 %)	1	108	(100 %)	1	0	205(3)
135	54	(40 %)	1	81	(100 %)	1	0	209(5)
135	64	(47 %)	1	71	(100 %)	1	0	210(6)

Table 4.1: Performance of the extended BitBallot protocol. Using *simgrid-bitballot*, the protocol has been simulated for $n = 135$ peers and different configurations for the number of present peers n_{present} and non-colluding dishonest peers $n_{\text{dishonest}}$. The statistics take only the results of honest peers into account. # a_R denotes the number of distinct root aggregates within the set of all honest peers and $f_{\max}$ of a_R denotes the highest occurrence of one distinct a_R. The ratio among all honest peers is provided next to it. The relative root aggregate counter $\gamma_{a_R} = c_{a_R}/n_{\text{present}}$ is the number of initial aggregates in a_R normed to n_{present}. The relative error $\gamma^*_{a_R} = c^*_{a_R}/n_{\text{present}}$ is the number of manipulated initial aggregates $c^*_{a_R}$ in a_R normed to n_{present}. The column # REQ contains the number of requests of each peer up to the latest observed change of their root aggregate.

aggregation converges for any number of absent peers. Further, the aggregation converges with no absent peers and a honest majority. This behaviour supports the identified *eventually accurate* property. Though not covered previously, also test cases with both absent and dishonest peers and less than 50 % honest peers (cf. line 7 in Table 4.1) allow for a convergence. The highest occurrence of one distinct a_R among all honest peers assumes only values close to the boundaries 0 % and 100 percent. That means, if the criteria for convergence are not met, the aggregation tends to diverge entirely.

Note that the number of requests # REQ per peer until local convergence is in all test cases in the order of about 220 requests and does

therefore not depend significantly on the number of present peers. This comes to a surprise and may be the result of an insufficient routing algorithm that dominates the number of requests, cf. Section 4.5.2.

Next, the correctness of the simulated aggregation is analysed. A root aggregate a_R of a peer P_i is called correct if it equals the $\oplus$ aggregation of all n_{present} present peers' initial aggregates. Missing initial aggregates and manipulated or erroneous aggregates in a_R diminish the accuracy, respectively, the integrity, and can be quantified. An initial aggregate that is manipulated after having been aggregated is counted as one missing initial aggregate and a manipulated aggregate of weight one. The aggregation is called correct if all honest peers have computed an accurate a_R, which can be deduced from the average value and standard deviation of three parameters. The relative initial aggregate counter γ_{a_R} of the root aggregate normed to the number of present peers n_{present} allows to assess the completeness. All manipulated aggregates a^* of dishonest peers are tagged in the simulation and their initial aggregate counters c_a^* are summed up separately in parallel to c_a throughout the simulation to compute eventually the relative error $\gamma_{a_R}^* = c_{a_R}^*/n_{\text{present}}$ of the root aggregate a_R. The third parameter is the already introduced number of distinct root aggregates $\#\, a_R$. So if $\gamma_{a_R} = \#a_R = 1$ and $\gamma_{a_R}^* = 0$, then the aggregation must be correct, because all peers have computed the identical a_R and implicitly validated that their initial aggregate has been included. This is the case for all configurations in Table 4.1 that meet the assumption of either a minority of dishonest peers and no absent peers, or vice versa, and confirms our previous findings. Configurations with very few honest peers exhibit significantly deviating values for γ_{a_R} and $\gamma_{a_R}^*$, which indicates that the aggregation did not converge to a correct root aggregate. It seems that the aggregation gets steadily more correct as the configuration approaches to fulfil the assumptions.

4.8 SUMMARY

The work on the implementation has lead to multiple insights. The absence of peers and potentially incomplete or erroneous aggregate containers require peers to request permanently aggregates for all ancestor nodes. Every time a request is legitimate, information on the tree position of both requester and responder is leaked reciprocally. After the aggregation, most peers have correctly assigned nearly all other peers to their respective relative sibling subtree. That means also, that all n peers contacted all other peers.

Global deadlines were introduced, so that peers may discover the aggregate containers of all represented sibling nodes first and spread in consequence less often incomplete containers. The need of corrections decreases and subsequently the amount of required requests. However, the determination of an optimal deadline has not been straight

forward. A too small value leads to massive additional requests due to corrections. A too large value leads to many redundant requests and the overall number of requests becomes eventually independent of the number of peers as it has been observed in Table 4.1. An improvement can be expected if deadlines for parent aggregates of nodes with depth d are computed in function of d. Especially for large d, i. e. few sibling peers, the peer discovery requires many requests. Hence, a prevailing issue for large-scale simulations is the peer discovery, that has been left so far unchanged with regard to the original BitBallot protocol.

Even though not experimentally verified, further issues may arise as soon as the dishonest peers do not provide randomly manipulated aggregate containers, but attack instead the mechanism of the clustering and correction algorithm. For instance, dishonest peers may construct forged aggregate containers with child aggregate IDs of other clusters to provoke erroneous merges of clusters. So far, no counter strategy has been developed.

BitBallot and its extended versions assume a trusted authority. The original protocol incorporates separation of powers to three distinct authorities that are assumed to not collude. If this assumption does not prove true, and a honest but curious authority is assumed instead, the confidentiality diminishes drastically. The authority A is responsible to assign each peer P_i to a leaf node x_i of the tree. It can request the aggregate of the parent node of any x_i that contains the initial aggregate of a_i. Depending on the aggregation operation $\oplus$, it may be possible to distinguish all k aggregates. Then, A can link a_i to P_i with an uncertainty of $1/k$. The uncertainty decreases further if the peers associated to sibling leaf nodes of x_i are absent or if they have chosen identical initial aggregates.

The simulation results demonstrate that the influence of a dishonest minority of peers increases with decreasing number of present peers, or in voting terms, with decreasing voter turnout. With less than 50 % of present peers, already few percent of non-colluding dishonest peers may harm the convergence and the correctness.

5 ADVOKAT

I'm a big advocate of freedom: freedom of speech, freedom of expression, freedom of thought.

— Jimmy Wales (co-founder of Wikipedia)

The proposed extensions of BitBallot are insufficient to allow for large-scale aggregations. The convergence and the correctness of the protocol depend to a large extend on the ratio of present to registered peers. This is in particular an issue in the use case of votings with low voter turnout. Furthermore, a curious authority may break the confidentiality of the initial aggregates. The encountered difficulties lead to a reconsideration of the protocol's principle concepts. Thus, the *aggregation for distributed voting online using the Kademlia DHT*, ADVOKAT for short, has been conceived that is a novel distributed protocol for confidential aggregations (Riemann and Grumbach, 2017b).

Advokat: German for lawyer (dated)*; borrowed from Latin* advocatus *for i. a. 1. attendant (friend who supports in a trial) and 2. mediator.*

The design goals of ADVOKAT are presented hereafter. An introduction of its principle concepts follows in Section 5.2. In Section 5.3, the protocol is presented in detail. Then in Section 5.4, light is shed on the properties of the protocol. A report on its JavaScript implementation used for simulation purposes is given in Section 5.5. A summary concludes this chapter.

5.1 DESIGN GOALS

The design goals of BitBallot presented in Section 4.1 are adopted and complemented on the basis of the previously encountered limitations. Like BitBallot, the new protocol shall provide for confidentiality-preserving aggregation of initial aggregates from peers with no or little cryptography. No authority shall be required after the preparation phase. The amount of knowledge on initial aggregates that each peer acquires during the aggregation shall be bounded and kept as small as possible to lower the impact of potential leaks.

Different than BitBallot, the new protocol shall allow for efficient large-scale aggregations and must therefore provide an effective peer discovering algorithm. The protocol shall ensure a correctness that is as independent as possible of the ratio of present peers to account e. g. for votings with a low voter turnout. The confidentiality of initial aggregates shall no longer be harmed in case of a dishonest authority.

5.2 PRINCIPLE CONCEPTS

Distributed aggregation protocols, such as SPP (see Section 3.1.6.1), and the example of BitBallot demonstrate that tree-like structures offer the potential for efficient aggregations with a communication complexity of $O(\text{Poly}(\log n))$ for each peer (Gambs et al., 2011). This leads to two insights. First, BitBallot's aggregation over a tree is a promising concept. Each peer is assigned to one distinct leaf node of an aggregation tree whose structure determines the aggregation. Intermediate aggregates of each node are computed based on the aggregates of its child nodes. Second, the peer discovery may also benefit from an underlying tree structure. Hence, the new concepts of ADVOKAT introduced in the following concern at foremost the overlay network.

Several concepts of BitBallot are adopted with no or small modifications. The *aggregation algebra* presented in Section 4.2.3 ensures not only compatibility of the protocol with different voting systems, but further allows for various other applications.

The concept of the *aggregate container* of an aggregate introduced in Section 4.3 and extended in Section 4.6 is also maintained. The container consists of the aggregate a of a subtree $\widehat{\mathbb{S}}(x, d)$ and associated properties of a, also called *aggregate metadata*, whose selection depends on the protocol extension (see Section 5.3). As in Bitballot, the *aggregate ID* is defined as the hash of its aggregate container, so that in practice no metadata can be manipulated without changing the aggregate ID.

The dissemination of aggregates is realised by pull gossiping for the same reasons as described in Section 4.2.1. Though, the secrecy mechanism of the pull principle, that is based on peers responding to requests with one computed or acquired aggregate of any child node, is abandoned.

5.2.1 *Peer Discovery provided by Kademlia*

BitBallot employs a peer discovery technique like early versions of the distributed file sharing protocol BitTorrent covered in Section 3.2.2.1. The BitTorrent tracker provides peers with a random set of other peers subscribed to the same torrent file, and the BitBallot tracker does likewise for those subscribed to the same aggregation. Further, both protocols allow to discover new peers due to received remote procedure calls (RPCs) from yet unknown peers. In the course of time, all BitTorrent peers may eventually accumulate the same file, so that peers may eventually receive file pieces from any other peer. This is fundamentally different from BitBallot whose peers hold only partial knowledge of the aggregation and depend on the discovery of those peers that possess relevant information, i. e. aggregates of sibling nodes.

To reduce the load on BitTorrent trackers, Loewenstern (2008) suggested in the BitTorrent enhancement proposal No. 5 to use the DHT

Kademlia (see Section 3.2.1.1) to distribute the peer discovery during an ongoing file transfer among all peers. That means, peers need only to contact the tracker to find a first Kademlia peer in order to connect to the DHT. Then, other peers can be found in the DHT and the tracker is not required any longer. It is reasonable to follow this evolution of BitTorrent and use the same peer discovery mechanism for ADVOKAT. The efficient discovery of peers in sibling subtrees is covered hereafter.

5.2.2 *Aggregation over a Binary Tree*

Consider that each peer P_i with an aggregation tree leaf node x_i is connected to the Kademlia DHT with KID $\check{x}_i$. Then, peers can easily discover new peers by sending `FIND_NODE`$(\check{x}_j)$ RPCs with an arbitrary $\check{x}_j$, which delivers a set of peers with KIDs close to $\check{x}_j$ in terms of the Kademlia distance (see Section 3.2.1.1). However, peers close to KID $\check{x}_j$ do not have a specific sibling relation with node x_j in the aggregation tree. Consequently, `FIND_NODE` RPCs cannot be used here to find efficiently peers associated to sibling nodes. Like in BitBallot, a number of RPCs in the order of $\mathcal{O}(n)$ is necessary to discover a peer assigned to a sibling leaf node of x_i.

A simple solution to reduce the message complexity is to require for KIDs $\check{x}_i = x_i$. The tree overlay used for the aggregation becomes congruent with the Kademlia tree overlay. That means that the routing table of Kademlia, a random subset of peers belonging to sibling subtrees $\mathbb{S}(x_i, d)$ for each depth $d = h, \ldots, 1$, is reusable to send requests for intermediate aggregates of the respective sibling subtrees. That means further that the aggregation tree must be a binary tree with arity $k = 2$ to match Kademlia's underlying tree. Kademlia's XOR distance function d (cf. Section 3.2.1.1) is a valid metric and unidirectional. That means, for any given node $\check{x}_1$ and distance $\Delta > 0$, there is exactly one other node $\check{x}_2$ such that $d(\check{x}_1, \check{x}_2) = \Delta$. In order to maintain the properties of the Kademlia tree, its arity cannot be changed to allow for an underlying tree with $k > 2$ as required by BitBallot.

A fixed arity of 2 has a direct impact on the secrecy and scalability. Peers cannot reply any more to requests with a random sibling aggregate to destroy the link between their response and the aggregate of their node. Requester and responder can with certainty distinguish between their aggregate and other aggregates of the only sibling node. So while in BitBallot peers do not learn with certainty the source P_i of an acquired initial aggregate a_j, there is no ambiguity in ADVOKAT and the source is always $P_i = P_j$. Hence, ADVOKAT employs pull RPCs different from BitBallot to request aggregates. In BitBallot, peers send RPCs specifying a parent node N. Valid responses are the aggregates of any child node of N. Instead, ADVOKAT employs RPC requests specifying a node N and only aggregates of N are valid responses. Consequently, peers can acquire precisely those aggregates

that are required for their computation. In the model with only honest peers, at most h requests with tree height h are required to compute the root aggregate. For this, sources of aggregates of a given node N must be known in advance, which is granted due to Kademlia (cf. Section 5.2.1). The performance drawback of Bitballot due to redundant responses with already known aggregates presented in Section 4.5.2 is eliminated. The impact of an arity $k = 2$ on the protocol's properties is discussed in detail in Section 5.4.

5.2.3 *Distributed Tree Configuration*

The BitBallot protocol provides for an authority A entrusted to assign every peer P_i to a random distinct leaf node ID x_i. In the course of this process, A gains the capacity to find out x_i for any given P_i and vice versa, which can be used for targeted attacks. In the adversary model of a dishonest authority, A can compromise both correctness and confidentiality as described in Section 4.5.3.

The basic Kademlia protocol as introduced in Section 3.2.1.1 provides for KIDs x_i chosen by the peers on their own discretion. In this case, dishonest peers may choose a KID very close to those of their target peer in order to obtain with high probability its initial aggregate. A Sybil attack with the target's KID may allow to provide an initial aggregate on behalf of it. To minimize the risk of such targeted attacks, P_i must be assigned to a random KID x_i, that is verifiably random.

For this, a KID distribution scheme is employed that is similar to the proposal from Baumgart and Mies (2007) presented in Section 3.2.1.1, of the one part, and to Anonymous Voting recalled in Section 3.1.2, of the other part. Similar to BitBallot's extension to account for dishonest peers presented in Section 4.6.2.1, P_i and A generate jointly an *authorisation token* t_i for P_i. A blind signature scheme prevents that A can establish a link between P_i and t_i. Whereas Baumgart and Mies suggest to derive the KID $x_i = \eta(pk_i)$ by hashing the public key pk_i, ADVOKAT uses $x_i = \eta(t_i)$. As a result, x_i is neither controlled solely by A or P_i and can be verified with t_i and pk_i by other peers. Even if both A and P_i are dishonest and collude, a hash collusion must be found to choose a given x_i, which requires computational expensive proof of work.

Note that the described scheme to generate KIDs cannot exclude that two distinct peers P_i and P_j with or without distinct secret keys sk_i, respectively sk_j, compute identical KIDs $x_i = x_j$ due to a hash collision. To reduce the chance of hash collisions, the size of t_i and x_i in bits must be chosen sufficiently large, so that collisions of KIDs are very unlikely and can be neglected. In the rare event of a collision, affected peers may not be able to contribute to the root aggregate.

The size of x_i in bits translates directly to the height h of the aggregation tree. In contrast to BitBallot, the height of the aggregation tree is

independent of the total number of peers n. That means further that the tree in ADVOKAT is generally very sparsely populated.

The verifiable and secret KIDs allow further to provide a proof of legitimacy of aggregate requests. Request senders have to sign their message using the secret key sk_i and provide pk_i and t_i along with every message. A request concerning node N is only responded if the message signature and the public key can be verified and the request sender actually needs the aggregate to compute the root aggregate. Note that a dishonest authority does not have the capacity to forge such a proof due to unknown secret sk_i. To protect against leaks of aggregates due to overhearing of the communication channels among peers, confidential channels can be established using the key pairs (pk_i, sk_i) and a communication security protocol such as transport layer security (TLS).

5.2.4 *Consensus on Intermediate Aggregates*

BitBallot and ADVOKAT do not use cryptography to enforce correctness as it is the case with most classical online voting protocols such as Helios (see Section 3.1.4.1). Honest peers compute intermediate aggregates using the aggregation algebra and trust is assumed in the correctness of acquired foreign aggregates. BitBallot uses further a symmetric cipher to limit the manipulations of dishonest peers locally. The ADVOKAT proof of legitimacy introduced above prevents in a similar way that dishonest peers manipulate foreign aggregates. However, the aggregation can still diverge if dishonest peers manipulate their own aggregates. To allow for a converging aggregation despite absent or dishonest peers, the correction algorithm developed as part of a BitBallot extension in Section 4.6.2 is integrated in ADVOKAT and further strengthened. Likewise, it must be assumed that the majority of peers is honest.

As pointed out previously, the computation of intermediate aggregates is redundant. With only honest peers, the number of sources of an aggregate a_N of node N equals its initial aggregate counter c_{a_N}. If in the adversary model of absent and dishonest peers incomplete or erroneous aggregates of the same node N emerge, a consensus must be established among all honest peers on which aggregate shall be used to compute the aggregate of the parent node of N. According to Property 4, the intermediate aggregate a with the most sources is assumed to be the most accurate, because every source computed the identical aggregate a and implicitly verified that it is based on its initial aggregate. The number of sources per aggregate is not easily accessible in a distributed network with no public bulletin board (PBB) and may even change in time as more peers compute a or exchange a for a different aggregate. Further, it must be ensured that dishonest peers cannot employ fake sources to promote arbitrary aggregates.

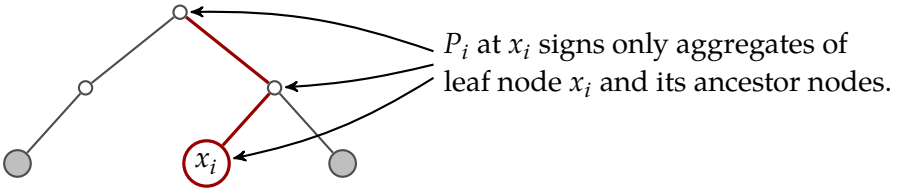
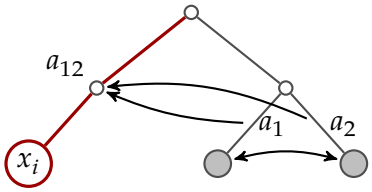
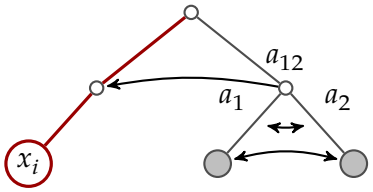


Figure 5.1: Eligibility of signatures in ADVOKAT. The public key pk_i of P_i is tied by its authorisation token t_i to one leaf node $x_i = \eta(t_i)$. Signatures of P_i are only valid for aggregates of node x_i and its ancestor nodes.



(a) The consensus is only identified externally.



(b) The consensus is already identified internally.

Figure 5.2: Consensus on aggregates in ADVOKAT. In both scenarios, two sibling peers (marked in grey) request the aggregate of their sibling node and compute the parent aggregate a_1, respectively a_2. In (a), only the peer at x_i in the sibling subtree identifies the consensus after having received the parent aggregates a_1 and a_2 and found them to be equal, i. e. $a_{12} = a_1 = a_2$. The two sources of a_{12} are unaware of $a_1 = a_2$. In (b), the consensus is identified internally. Due to reciprocal requests of the parent aggregate, the two sources find $a_{12} = a_1 = a_2$. Signatures allow them to provide a third peer x_i with a verifiable proof of consensus on a_{12}.

Peers employ sampling in order to approximate the relative number of sources of competing aggregates of a node N. For this, *confirmation requests* specifying a node N are sent to random peers in the subtree of N to acquire their aggregate of N. Remember that pull requests allow a peer P_i to acquire aggregates of sibling nodes of x_i and its ancestors, whereas confirmation requests are employed to acquire aggregates of x_i and its ancestors.

Confirmation requests continue until a predefined level of certainty is reached. Fake sources can be efficiently excluded using verifiable signatures on aggregate IDs, *aggregate signatures* for short, that are linked to the source P_i by its public key pk_i. The eligibility of P_i to sign aggregates a_N of node N is proven by the authorisation token t_i. The token t_i is bound to one leaf node $x_i = \eta(t_i)$ and signatures from P_i shall only be valid if $N = x_i$ or an ancestor node of x_i. See Figure 5.1 for an illustration.

As shown in Figure 5.2, the number of redundant requests can be decreased if more than one aggregate signature is shared with each

aggregate container. For this, peers in the subtree with root at node N compute first the common parent aggregate of N, which implies reciprocal child aggregate requests. New is then that peers, once the parent aggregate is computed, continue to request random other peers in the subtree for their subtree in order to acquire further signatures of their parent aggregate. At this stage, a peer may also discover that its parent aggregates is diverging. In this case, attempts to correct potential erroneous aggregates can start before erroneous aggregates are further disseminated to peers that are not in the subtree. If the majority of sampled peers in the subtree computed the identical aggregate of N, the parent aggregate can be *confirmed* and is used to respond to requests for aggregates of N. Acquired signatures are sent along to proof the consensus.

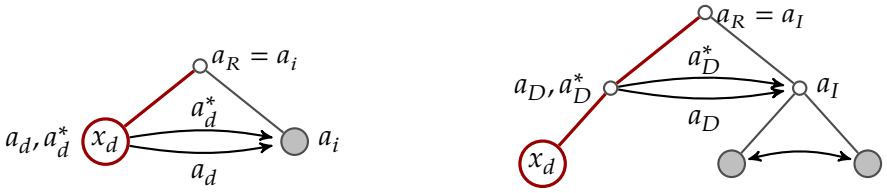
In the adversary model of colluding dishonest peers, constraints on the position in the tree of the signing peers with respect to the tree configuration must be imposed. Otherwise, dishonest peers may exclusively provide only signatures of colluding peers to pretend that a common manipulated aggregate represents the consensus in the subtree.

If for some reason some mandatory signatures cannot be acquired from the relevant peers, honest peers provide with their responses the child aggregate containers with their mandatory signatures, so that the aggregate container can be computed locally and compared for verification. The negative impact on confidentiality is accepted in favour of an improved correctness and convergence.

5.2.5 *Blocking Dishonest Peers*

The aggregate signatures introduced in ADVOKAT above are not only used to find a majority in case of conflicting aggregates of the same node, but provide also a mean to detect protocol deviations by dishonest peers. The principle idea inspired by game theory is to create an incentive for dishonest peers P_d to not deviate from the protocol. Protocol deviations of P_d may be detected with a given probability by honest peers due to P_d's signatures on conflicting aggregate containers. In the case that a honest peer P_i detects a protocol deviation of P_d, P_i invalidates locally its authorisation token t_d. Hence, P_d's signatures become invalid and in consequence its aggregate requests are not responded any more. P_i stores further the *proof of deviation* in form of signatures and the aggregate containers with or without the aggregates in the distributed hash table (DHT) Kademlia, so that other peers may find this information with a lookup in the DHT for the token t_d. Peers shall only accept and relay verified proofs in the DHT to prevent misuse and manipulations of the DHT.

All peers use sporadically DHT lookups for a random subset of foreign tokens learned from responses to aggregate requests. If such a



(a) Initial aggregates

(b) Intermediate aggregates with a counter of $c = 1$

Figure 5.3: Detecting dishonest peers in ADVOKAT. Two scenarios are considered with honest peers computing aggregates a_i, respectively a_I, and one dishonest peer P_d at leaf node x_d. Arrows indicate pull requests. In can be shown that in all considered scenarios, attempts of P_d to introduce manipulated initial aggregates a_d^* or intermediate aggregates a_D^* can be detected.

lookup reveals a proof of deviation, the token is again invalidated with the already mentioned consequences. In order to increase the performance, honest peers may also provide the proof along with responses to related aggregate requests.

For illustration purposes, consider the scenarios with different subtree configurations in Figure 5.3 with their local root node denoted R. The configuration in Figure 5.3a consists of a small tree with only two peers associated to sibling leaf nodes. A honest peer P_i requests two times[1] P_d to get its initial aggregate. P_d is dishonest and provides two distinct signed initial aggregates a_d, a_d^*. Thus, P_d deviated from the protocol, which provides only for one valid initial aggregate per peer.

P_i notices the two initial aggregates signed with the same key pk_d and invalidates in consequence the authorisation token of P_d and stores the proof of deviation with the signatures of a_d, a_d^* in the DHT. Further, P_i discards both a_d, a_d^* and computes the parent aggregate (here the root aggregate) $a_R = a_i$ without any initial aggregate from P_d. If P_i receives a request for the root aggregate, it may choose to include the proof of derivation in its response to allow for an efficient verification of the aggregate, that comprises one initial aggregate container less than it is expected for the tree configuration. Note, that peers P_i can assess the tree configuration close to their leaf node x_i thanks to the Kademlia routing tables (cf. Section 3.2.1.1).

A proof of deviation may require only one initial aggregate a_d, if a_d or its aggregate container is invalid, e. g. due to a counter $c_{a_d} \neq 1$, or in the case of an election, due to an invalid combination of chosen options. So if P_d provides such an invalid initial aggregate a_d, then the aggregate a_d, its container and the aggregate signature constitute a verifiable proof

1 Note that P_i does not send a pull request to P_d a second time after it has already received one valid initial aggregate if no prior aggregation issue is detected. The provided simplified example illustrates the principle idea that is then applied to more complex cases.

of deviation, which is also stored in the DHT. Note that in ADVOKAT, initial aggregates of provably dishonest peers are not confidential.

P_d may deviate from the protocol by a computation of a manipulated parent aggregate. Both P_d and P_i learn about their deviating parent aggregates in the course of confirmation requests. A proof of deviation is possible here and consists of both initial aggregates and the manipulated parent aggregate. Depending on the use case of ADVOKAT, P_i may store the proof in the DHT and sacrifice the confidentiality of a_i up to its public key pk_i, which is a pseudonym of P_i. Alternatively, P_i may only provide the proof as part of its response to requests until a local majority of peers confirmed and used the accurate parent aggregate computed by P_i.

In summary, a manipulation of initial aggregates by P_d with a honest peer associated to its sibling leaf node leads with certainty to an exclusion of P_d from the entire aggregation. P_d's initial aggregates are revealed to the public and are not included in the root aggregate. Hence, there is a strong incentive to respect the protocol at this stage.

In the scenario depicted in Figure 5.3b, a dishonest peer P_d is the only peer in a subtree with two honest peers in the sibling subtree, of which one peer P_i computes the intermediate aggregate a_I. The local root is denoted R. P_d computes without the need of prior aggregate requests the parent agregate a_D. With its initial aggregate counter $c = 1$ of a_D, it is equivalent to an initial aggregate. Suppose P_d provides two distinct aggregates a_D, a_D^* to respond differently to pull requests from the honest peers in the sibling subtree, so that they compute distinct root aggregates. If for a confirmation of a_R computed by P_i at least one more signature from a peer in each child subtree of the root node is required, then P_i detects the deviation with certainty due to a confirmation request to the other honest peer.

To allow here for a proof of deviation, aggregate containers of aggregates a must be extended by the set H_a of its child aggregate IDs, which has been proposed first in Section 4.6.1 to cluster aggregates by their child branch. Here, the child branch is obvious without an analysis of H_a due to the fact that $x_i = \eta(t_i)$ with t_i provided along with each aggregate container. However, H_a is useful to detect protocol deviations if a verifiable signature is provided for each child aggregate ID in H_a. Hence, ADVOKAT is extended accordingly.

With signatures on child aggregate IDs, honest peers in the scenario above can provide a proof of deviation consisting of one signature of a_D acquired by a pull request and a second signature of child a_D^* acquired by a confirmation request. The erroneous a_R is recomputed without an aggregate of P_d and can eventually be confirmed. Having the proof of deviation of P_d, its signatures are not mandatory any more to confirm aggregates.

5.3 PROTOCOL DESCRIPTION

The protocol relies on peers P_i, an authority A and at least one tracker T. The authority is entrusted to certify the eligibility of peers. For this purpose, an authenticated, tamper-resistant communication channel between the authority and each peer is assumed, e.g. using an existing public key infrastructure.

Once certified to be eligible, peers join the DHT Kademlia that is mainly used to find other peers, but allows further to store and look up proofs of deviation. Similar to BitTorrent, a tracker is employed to provide an initial peer as an entry point. Peers communicate via pairwise channels with no security requirements.

The notation introduced for BitBallot and its extensions is adopted.

A	Authority
P_i	Peer, i-th out of n
a_i	Initial aggregate of peer P_i
(pk_i, sk_i)	Public and private key pair of peer P_i
$\eta(m)$	Hashing technique for message m, e.g. SHA-3
$\sigma_i(m)$	Peer P_i's signature scheme using (pk_i, sk_i)
$\sigma_A(m)$	Authority's signature scheme
$\chi(m, r)$	Blinding technique for message m and random number r
$\delta(s, r)$	Retrieving technique of blind signature

The notation used to describe positions in the tree is adopted from the presentation of Kademlia in Section 3.2.1.1. Remember that the subtree notation is overloaded to designate next to subtrees also the set of peers assigned to leaf nodes of the corresponding subtree.

k	Maximum number of contacts per tree segment (k-bucket)
x	Kademlia leaf node ID (KID) of size B
B	Size of a KID in bits, e. g. 160
x_i	KID of peer P_i
d	Node depth, i.e. number of edges from the node to the root
$\widehat{\mathbb{S}}(x, d)$	Subtree whose root is at depth d which contains leaf node x
$\mathbb{S}(x, d)$	*Sibling subtree* whose root is the sibling of the root of $\widehat{\mathbb{S}}(x, d)$

The aggregation algorithm relies on aggregate metadata, which are in general not directly involved in the aggregate computation. The metadata of an aggregate a together with a constitute the aggregate container of a with its properties listed hereafter:

h	Aggregate ID, hash $\eta(\cdot)$ of the aggregate container without h

a	Aggregate
c	Counter of initial aggregates in a, $c = c_1 + c_2$
c_1, c_2	Counter of initial aggregates of child aggregates
h_1, h_2	Aggregate ID of child aggregates
$\widehat{\mathbb{S}}(x, d)$	Identifier of subtree whose leaves' initial aggregates are in a

Similarly to the aggregation of aggregates, one or two aggregate containers of a_1, a_2 can be aggregated to a parent aggregate container. To inherit the commutativity of the aggregation of aggregates $\oplus$, the order of c_1, c_2 and h_1, h_2 is crucial and must be sorted with respect to e. g. the node position of their aggregate, left (0) or right (1), or by the ID of their aggregate.

The structural steps of the proposed aggregation protocol follow:

PREPARATION PHASE Peers create personal public and private key pairs and send authorisation requests with their blinded public key to the authority. Once for each peer, the authority signs the peer's blinded public key without learning it and sends the signature to the peer. Supported by the tracker, peers join the tree-like overlay network of Kademlia.

AGGREGATION Peers assign their *initial aggregate* to their leaf node and compute collectively the *root aggregate* from all initial aggregates using the *distributed aggregation algorithm*. This requires the computation of *intermediate aggregates* of all their ancestor nodes in the Kademlia tree.

EVALUATION On fulfilment of a well-defined verification criteria, peers accept their root aggregate as *final root aggregate*. The outcome (e.g. election result) is eventually derived from the final root aggregate.

A detailed description of all steps follows. The resulting properties are then discussed in Section 5.4.

5.3.1 *Preparation Phase*

The authority A and every peer P_i create at their discretion a key pair (pk_A, sk_A), respectively (pk_i, sk_i). The authority establishes further a set $\mathbb{P}$ of all peers that are eligible to contribute their initial aggregates to the aggregation. Eventually, all peers are notified by the authority of the upcoming aggregation and provided with:

- the public key pk_A of the authority;
- a deadline for the preparation phase;

- at least one tracker T to discover other peers;
- the set of valid initial aggregates to choose their own initial aggregate a_i and verify initial aggregates a_j of other peers P_j.

Before the deadline of the preparation phase, each peer P_i must acquire its authorisation token t_i from the authority A in order to compute its Kademlia leaf node ID x_i. The following steps implement the distributed tree configuration presented in Section 5.2.3.

1. P_i uses the blinding technique with a random secret r_i to blind its public key to $b_i = \chi(pk_i, r_i)$.
2. P_i sends an *authorisation request* with b_i to A using the authenticated communication channel.
3. A denies the request if a prior request of P_i has been responded. Otherwise, A creates a signature $s_i = \sigma_A(b_i)$ and responds the request with it.
4. P_i unblinds the acquired signature s_i with its secret r_i to get the authorisation token $t_i = \delta(s_i, r_i)$.
5. Then, P_i determines its leaf node with KID $x_i = \eta(t_i)$ and connects to the Kademlia DHT using an already connected contact provided by one tracker T.

Here, the number of public keys pk_i with authorisation token t_i is limited by the authority A to one per eligible peer P_i (Gambs et al., 2011). As in FOO (Fujioka, Okamoto and Ohta, 1993), a blind signature scheme (Chaum, 1983) is used to ensure that A cannot link a given authorisation token t_i to its peer P_i. A does not intervene any further in the protocol once all eligible peers have acquired their authorisation token or the preparation phase is over. The setup of the DHT with peers P_i associated to leaf nodes x_i is carried out jointly according to the Kademlia protocol (Maymounkov and Mazieres, 2002).

The channel among peers is assumed to have no authentication or protection from overhearing. For this reason, all messages among peers, no matter whether part of Kademlia or specific to ADVOKAT, are signed. The message sender P_i provides further pk_i and t_i for signature verification. Messages without valid signatures are discarded.

If the eligibility to get a response on a given remote procedure calls (RPCs) is constrained by the tree configuration, as it is the case for pull and confirmation requests, the verified leaf node $x_i = \eta(pk_i)$ of the request sender is used for an assessment. To prevent overhearing, responses are generally encrypted with an asymmetric cryptography cipher using pk_i of the request sender.

The public key and the authorisation token of peers are stored as part of the contact information in the DHT routing table. The response to

Kademlia's `FIND_NODE`(KID) RPC to lookup peers contains consequently for each peer P_i also pk_i and t_i for verification of eligibility.

Is is assumed that by the end of the preparation phase, all peers joint successfully the Kademlia DHT. Thus, every peer P_i has knowledge about its closest peers, i. e. peers P_j associated to x_j with a small distance $d(x_i, x_j)$ as defined in Section 3.2.1.1. For each sibling subtree $\mathbb{S}(x_i, d)$ with any depth $d \in \{B, \dots, 1\}$, P_i obtained up to k randomly chosen peers $P_j \in \mathbb{S}(x_i, d)$. The obtained set of peers is exhaustive for subtrees $\mathbb{S}(x_i, d)$ with not more than k peers.

5.3.2 *Aggregation Phase*

During the aggregation phase, all peers run the *distributed aggregation algorithm*, that is presented hereafter in two steps. Similar to the aggregation in BitBallot (cf. Section 4.3.2), every peer P_i with KID x_i computes the intermediate aggregate a of each of its subtrees $\widehat{\mathbb{S}}(x_i, d)$ with depth $d = B - 1, \dots, 0$ and assigns a to the root node of the respective subtree. As discussed in Section 5.2.1, the aggregation is carried out along the tree overlay network of the Kadmelia DHT, so that the sources of sibling subtree aggregates can be found in the Kademlia routing table.

BASE STEP First, every peer P_i assigns its initial aggregate a_i to its leaf node x_i. For this, an aggregate container must be provided whose initial aggregate counter is $c_{a_i} = 1$. With no child nodes of x_i, the properties related to child nodes, i. e. the child aggregate IDs h_1, h_2 and counters c_1, c_2 are unset. The container is associated to the subtree $\widehat{\mathbb{S}}(x_i, d)$ with $d = B$, respectively to the node that is the root of this subtree, which is here identified by the leaf node x_i and the depth of the subtrees's root node.

h	Aggregate ID, hash $\eta(a = a_i, c = 1, \widehat{\mathbb{S}}(x_i, B))$
a_i	Initial aggregate of P_i
c_i	Counter of initial aggregates $c = 1$
c_1, c_2	Counter of initial aggregates of child aggregates *not set*
h_1, h_2	Aggregate ID of child aggregates *not set*
$\widehat{\mathbb{S}}(x_i, d)$	Identifier of subtree to which a is associated

P_i creates then the *aggregate signature* of a_i. Anticipating the needs to identify protocol deviations as presented in Section 5.2.5, the signatures of any aggregates a are constructed in a manner to allow their verification with only aggregate ID h, counter c and depth d given. Consequently, the aggregate signature is computed by its source P_i with $\sigma_i(h, c, d)$ and may be denoted $\sigma_i(a)$ for short. During the base step, P_i computes the signature $\sigma_i(a_i)$ of its inital aggregate.

As in BitBallot, the distributed protocol ADVOKAT does not provide for a precise casting step in which a_i is submitted to an authority. However, the sharing of a_i with $\sigma_i(a_i)$ and its subsequent signed parent aggregates in the next step fulfils a similar function. The submission of a_i by P_i is witnessed and due to its signature $\sigma_i(a_i)$ provable to third parties.

RECURSIVE STEP The recursive step involves the exchange of aggregates among peers using pull and confirm RPCs. P_i executes this step recursively with a parameter depth $d \in \{B-1, \dots, 0\}$ to compute or correct the intermediate aggregate of the subtree $\widehat{\mathbb{S}}(x_i, d)$ until the root aggregate has been computed and confirmed with a given certainty. The progress of the aggregation of all peers is loosely synchronised, because peers may have to wait for intermediate aggregates to be computed in order to continue their computation.

During the base step of P_i, the aggregate a_i was associated to the leaf node x_i with depth $d = B$. Hence, the recursive step must be first carried out with $d = B-1$ to compute the subsequent parent aggregate. The procedure of the recursive step calls recursively itself with a depth of either $d - 1$ to compute the subsequent parent aggregate or $d + 1$ to correct a previously computed child aggregate. The recursion stops when the root aggregate a_R of the root node R with $d = 0$, or of $\widehat{\mathbb{S}}(x_i, 0)$ in terms of subtrees, has been computed and confirmed.

To compute the parent aggregate of $\widehat{\mathbb{S}}(x_i, d)$ during the recursive step with depth d, the child aggregates of $\widehat{\mathbb{S}}(x_i, d+1)$ and $\mathbb{S}(x_i, d+1)$ are required. While P_i is a source of aggregates of $\widehat{\mathbb{S}}(x_i, d+1)$, the foreign aggregate of the sibling subtree $\mathbb{S}(x_i, d+1)$ must be first acquired from any peer $P_j \in \mathbb{S}(x_i, d+1)$. Section 5.2.4 introduced the concept of *confirmed aggregates* that have been computed, and therefore validated and signed by multiple sources. P_i sends pull requests to repeatedly randomly chosen P_j until a confirmed aggregate has been acquired or an equivalent proof of accurate aggregation has been provided.

In Figure 5.4, the procedure of $P_j \in \mathbb{S}(x_i, d)$ is depicted to produce such a confirmed aggregate of $\mathbb{S}(x_i, d)$ that can be verified by $P_i \in \widehat{\mathbb{S}}(x_i, d)$. If $\mathbb{S}(x_j, d+1) \neq \emptyset$, P_j sends to repeatedly randomly chosen $P_l \in \mathbb{S}(x_j, d+1)$ pull requests in order to acquire a confirmed sibling aggregate of $\mathbb{S}(x_j, d+1)$ (①) or an equivalent proof of accurate aggregation. If after several attempts or a given time-out, no confirmed aggregate could be acquired or if $\mathbb{S}(x_j, d+1) = \emptyset$, no sibling aggregate is used in the following. Then, the so-called *aggregate candidate* a of $\mathbb{S}(x_i, d)$ is computed (②) with all obtained child aggregates. Subsequently, the aggregate container of a is computed. If only one child aggregate was used, h_2 is not set and $c_2 = 0$. P_j creates then an aggregate signature $\sigma_i(a)$.

Next, P_j sends confirmation requests to other peers in $\mathbb{S}(x_i, d)$ to acquire their signed aggregate candidate or if possible already confirmed

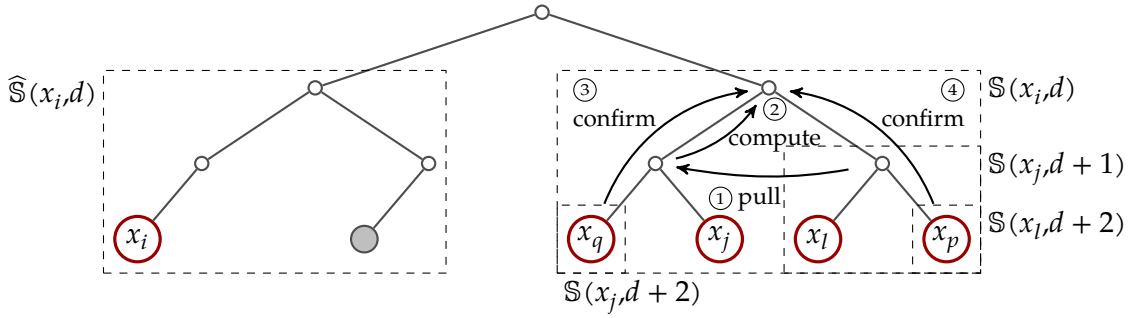


Figure 5.4: Pull and confirm of aggregates in ADVOKAT. P_j with x_j produces a confirmed aggregate container of $\mathbb{S}(x_i,d) = \widehat{\mathbb{S}}(x_j,d)$. This scheme applies to all tree levels with possibly large subtrees with multiple potential sources. The tree is sparse, so that $\mathbb{S}(x_j,d+2)$ or $\mathbb{S}(x_l,d+2)$ may be empty. If so, the depth is further increased until a non-empty subtree may be found. If all subtrees are empty, the request is omitted.

aggregate. In any case, P_j must obtain a specific set of aggregate signatures on the identical aggregate a that are later verified by P_i:

1. P_i requires the signature $\sigma_j(h,d,c)$ on container hash and counter.
2. If $c > 1$, there is at least one child aggregate with hash h_1 and counter c_1 and $\sigma_j(h_1,d+1,c_1)$ must be provided.
3. If $c > 1$ and $c_1 > 1$, a confirmation request (③) is necessary to provide $\sigma_q(h,d,c)$ from $P_q \in \mathbb{S}(x_j,d')$ with the smallest $d' > d+1$ for a non-empty subtree, ideally in the subtree $\mathbb{S}(x_j,d+2)$.
4. If $c > 1$ and $c_2 > 0$, P_j provides $\sigma_l(c_2,d+1,h_2)$ acquired previously (①) as 1. signature.
5. If $c > 1$ and $c_2 > 0$, a confirmation request (④) is necessary to provide $\sigma_l(h,d,c)$ if P_l for $c_2 = 1$, and otherwise $\sigma_p(h,d,c)$ from $P_p \in \mathbb{S}(x_l,d')$ with the smallest $d' > d+1$ for a non-empty subtree, ideally in $\mathbb{S}(x_l,d+2)$.

The number of distinct signatures on h, here 3 at most, is a security parameter denoted l. The parameter l may be increased for some applications to improve the accuracy at the expense of additional communication. In general, l may depend on the local tree configuration and the initial aggregate counter c.

Next to the signatures, P_i ensures further that the counter c does not exceed the number of peers in $\mathbb{S}(x_i,d)$ stored in the Kademlia routing table if $c < k$, i. e. the set of peers in $\mathbb{S}(x_i,d)$ exhaustive. For $c \geq k$, the set may be exhaustive or not.

If after several confirmation requests and within a given time limit, the required signatures cannot be obtained by P_j, P_j has to refrain from

its aggregate candidate a or provide an equivalent proof of accurate aggregation. In most cases, there is a non-zero chance that the confirmed child aggregates of the parent aggregate a are manipulated or erroneous. If the confirmation requests reveal to P_j that the majority of the requested sample has computed an aggregate candidate a' distinct from a or that one aggregate candidate a' with $c_{a'} > c_a$ and a given number of obtained signatures is more complete[2], P_j repeats the recursive step for $d-1$ in case of a distinct child aggregate of $\widehat{\mathbb{S}}(x_j, d+1)$ in order to correct or verify its previously computed child aggregate. If only the foreign child aggregate of $\mathbb{S}(x_j, d+1)$ is found to be distinct, P_j carries out the recursive step again for the same depth d.

If P_j can proof its accurate aggregation using aggregate signatures or if P_j's aggregate candidate a meets a given level of confidence computed on the basis of the sample size and the obtained majority, then P_j resorts to provide both child aggregates as a proof of accurate aggregation. Of course, both child aggregates must be either confirmed or come also with a proof of accurate aggregation. The proof of accurate aggregation allows eventually peers not in $\mathbb{S}(x_j, d)$ such as P_i to verify the aggregate a from P_j by repeating the computation of a and comparing aggregate IDs.

The aforementioned proof of accurate aggregation delivered by a peer P_j to P_i is based on one assumptions. Only information provided along the proof is considered. The proof of accurate aggregation of a signed parent aggregate a computed by P_j consists of the two child aggregates a_1, a_2 that must be each signed by as many distinct sources as there are initial aggregates included according to c_1, respectively c_2. That means, the child aggregates have been computed and signed by all potential sources. With increasing initial aggregate counters, such proofs become more and more difficult to deliver. For this reason, the proof of accurate aggregation is most important at the very early stage of the aggregation with subtrees of less than 4 peers. If, for instance, in a subtree with two peers one peer refuses to confirm the parent aggregate, the other peers can provide next to the not confirmed parent aggregate the two signed initial aggregates.

Note that the parent aggregate a may be not accurate despite its proof of accurate aggregation. This situation occurs inter alia when peers P_l with x_l join the aggregation very late and are only discovered after aggregates of common ancestor nodes have been computed and confirmed already. Here, P_j executes the recursive step again with the depth d of the first common ancestor node of x_j and x_l. Alternatively, a deadline that may depend on d may be defined to ignore late peer discoveries to the advantage of a more timely convergence of the aggregation.

2 The error correction is triggered by aggregate candidates with higher counters. Safeguard measures are considered in Section 5.6 to prevent exploits with manipulated aggregates.

Eventually, all peers compute the root aggregate candidate a_R of the root node of the entire tree. At this stage, different termination conditions may be used to confirm a_R with a higher scrutiny than previously demanded.

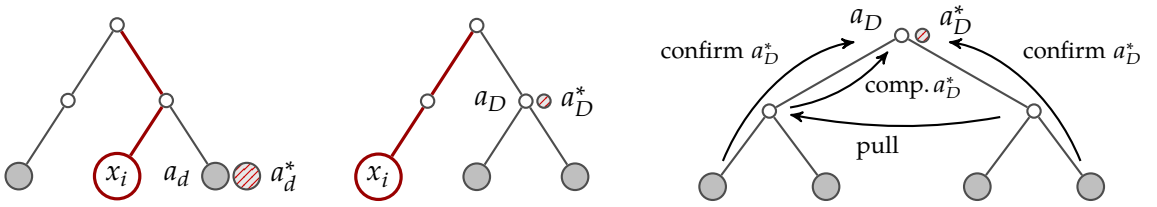
1. Confirmation requests continue until a given deadline has been passed.
2. Confirmation requests continue until a given ratio between aggregate signatures of a_R and its initial aggregate counter has been reached.
3. Confirmation requests continue until a given ratio between received signatures of a_R and those of any other root aggregate has been reached.

Of course, those termination conditions may also be combined. Note that a deadline or grace period after the aggregation terminated may be defined until whose expiration peers must continue to respond to confirmation requests. This is important to allow other peers to reach the termination conditions as well.

Next, the impact of dishonest peers is discussed. Unlike in BitBallot, the case of absent peers must not be addressed specifically. In the terminology of ADVOKAT, *absent peers* are those that obtained an authorisation token from the authority, but do not join the Kademlia DHT. The aggregation tree is sparse anyway, so that absent peers have no negative impact on the aggregation apart from their initial aggregates not being aggregated. The case of peers that join the DHT but do not respond to any requests is included in the following discussion concerned with arbitrary protocol deviations.

RECURSIVE STEP WITH DISHONEST PEERS In the remainder of this section, the detection of adversarial behaviour and the corresponding reaction of honest peers is detailed. The following procedures implement the concepts introduced in Section 5.2.5. The adversary model consists of Byzantine or *dishonest peers* that deviate arbitrarily from the protocol. Both cases of non-colluding and colluding dishonest peers are discussed. Further, it is assumed that the dishonest peers are in the minority among all present peers.

Throughout the aggregation, peers acquire with pull and confirmation request many signed aggregates. The acquired signatures are subject to a continuous analysis to identify signatures of say peer P_d that cannot occur if P_d complies with the protocol. Hence, P_d is recognised as a dishonest peer and a *proof of deviation* is assembled. The recursive construction of such proofs is presented in the following. For this, different subtrees $\widehat{\mathbb{S}}(x_d, d)$ with different tree configurations are considered. Here, a_d denotes one initial aggregate of $\widehat{\mathbb{S}}(x_d, d)$ with signature $\sigma_d(a_d) = \sigma(h_d, c, d)$ and the depth d. h_d is the aggregate ID of a_d



(a) Proof of 1st degree (b) Proof of 2nd degree (c) Confirmation of a manipulated aggregate a_D^* by 3 dishonest peers

Figure 5.5: Proof of deviations in ADVOKAT. Three tree configurations of subtrees are considered to introduce the mechanism allowing in some cases to proof a deviation from the protocol by one dishonest peer with certainty. The honest peer with KID x_i uses therefore conflicting signatures computed by one dishonest peer. Those initial aggregates a_d^* or intermediate aggregates a_D^* that create the conflict are marked with stripes. The configuration in (c) allows 3 dishonest peers to confirm a manipulated aggregate. A honest peer in the sibling subtree cannot detect the deviation with certainty.

and c is the initial aggregate counter. Alternative aggregates of the same subtree signed by P_d are denoted a_d^*. A confidentiality threshold parameter $c_{\min}^* > 1$ is introduced.

Suppose P_i with x_i in a tree configuration depicted in Figure 5.5a receives two distinct aggregates[3] with signatures $\sigma_d(a_d)$ and $\sigma_d(a_d^*)$ with $h_d \neq h_d^*$ and $c = 1$. This is the most basic form of provable deviation. The signed aggregates a_d and a_d^* are initial aggregates in the depicted case of $d = B$. Otherwise, they are equivalent to initial aggregates, because of $c = 1$, i. e. only one peer is involved in the aggregation and computation. The protocol does not allow peers to provide two distinct initial aggregates. Note that a correction may require to sign an updated aggregate, but those are only triggered in case of a diverging aggregation of multiple peers computing aggregates with $c > 1$. As only P_d has the key sk_d at it's disposal to compute the two signatures, P_d identified by its key pk_d must be dishonest with certainty.

The proof of deviation consists of both signatures and the necessary information for their verification. Note that the aggregate itself is not required, but only its aggregate ID, that is the hash of the entire aggregate container, the depth and the counter. As the proof relies directly on aggregates with $c = 1$, it is termed proof of first degree.

Once a proof of deviation of P_d has been assembled by a peer P_j, P_d is locally removed from the aggregation and the proof is published.

3 Note that P_i does not send a pull request to P_d a second time after it has already received one valid initial aggregate if no prior aggregation issue is detected. The provided simplified example illustrates the principle idea that is then applied to more complex cases.

1. P_j blacklists the key pk_d of P_d. Blacklisted keys are not used any longer to verify signatures.

2. P_j removes the P_d from its routing table. Peers with blacklisted keys are not added to the routing table later on.

3. P_j uses a `STORE` RPC to publish the proof in the DHT with the authorisation token t_d as key if no aggregates with $c < c^*_{\min}$ of peers are revealed that have not been proven dishonest.

4. If no ancestor aggregate of P_d's initial aggregate a_d has been confirmed by a local majority, P_j removes a_d from the aggregation and corrects already computed aggregates accordingly.

5. P_j attaches to responses on pull and confirmation requests for aggregates of $\widehat{\mathbb{S}}(x_i, d)$ with any d all known proofs of peers $P_d \in \widehat{\mathbb{S}}(x_i, d)$ if the initial aggregate counter c is smaller than k, the Kademlia k-bucket size, and no aggregates with $c < c^*_{\min}$ of peers are revealed that have not been proven dishonest. In addition, a random subset of limited size of those proofs with $c \geq c^*_{\min}$ is also attached.

It must be ensured that a dishonest peer P_d is not in charge to store proofs concerning itself, as otherwise also the lookup requests for those proofs could face manipulations. For instance, P_d could impede its dissemination. In order to store key/value pairs in the Kademlia DHT, keys are first hashed, thus $\eta(x_d) = \eta^2(t_d)$ with $x_d = \eta(t_d)$ is computed. The actual pair is then stored redundantly by multiple peers P_i with small Kademlia distance $d(x_i, \eta^2(t_d))$.

The second degree proof of deviation is based on the disregard of a first degree proof of deviation or a manipulated parent aggregate. Consider two distinct aggregates a_D and a^*_D with $c = 2$ and $h_D \neq h^*_D$ with signatures $\sigma_d(a_D)$ and $\sigma_d(a^*_D)$ as depicted in Figure 5.5b. If both aggregates have been computed on the basis of initial aggregates of the same set of two peers, which can be deduced from the mandatory child aggregate signatures, then P_d identified by pk_d deviated from the protocol with certainty, because it did not provide a first degree proof of deviation and did not remove aggregates of dishonest peers as described above. Though, no deviation is given if two distinct aggregates are based on initial aggregates of distinct sets of two peers, which may occur when dishonest peers are detected and their aggregates removed while missing aggregates are added of new discovered peers.

As mentioned in Section 5.2.5, P_d may also compute a manipulated aggregate a^*_D in disregard of the aggregation algebra and compute only one signature $\sigma_d(a^*_D)$. To proof the manipulation, the child aggregates corresponding to the signed child aggregate IDs are required, so that the aggregate can be computed by the verifier and compared to the signed manipulated a_D. By chance, some peers in sibling trees obtain

both child aggregates during the aggregation and can detect the manipulation. They publish the proof in the DHT if no aggregates with $c < c^*_{\text{min}}$ of peers are revealed that have not been proven dishonest. Otherwise, proofs are only attached to responses as described in 5. above. This kind of proof can be computed for parent aggregates of any depth d.

Note that first and second degree proofs do not require any assumptions weather dishonest peers collude or not. Though, to detect a deviation and prepare a proof, a honest peer associated to a sibling subtree is necessary. This is given with great probability by the random distribution of all peers to leaf nodes and the assumption of a minority of dishonest peers among all present peers and in each subtree.

Consider in the following the tree configuration of the subtree in Figure 5.5c. In a balanced subtree with 3 dishonest peers and one honest peer, a manipulated aggregate can be confirmed, if the dishonest peers use the signed child aggregate ID of the honest peer and compute all other required signatures on their own. For this, a collusion of dishonest peers must be assumed. Nonetheless, the honest peer may create of proof of deviation, but needs to reveal the child aggregate for this. Still, other peers may compute consecutive erroneous parent aggregates based on the confirmed manipulated aggregate without deviating from the protocol if they pull and confirm by chance only from the dishonest peers. For this reason, there is no proof of deviation of third degree. In sum, colluding dishonest peers can deviate in the unlikely event of a local majority of dishonest peers in a manner, that cannot be detected by honest peers with certainty. Manipulations are bound locally and are inherited by parent aggregates.

Peers do not attach proofs of deviations to responses in all cases. For this reason, the verification of signatures during the aggregation must be extended by one additional step. With a given probability, a public key acquired along with signatures is looked up by its authorisation token in the DHT to test weather a proof of deviation for it has been stored.

5.3.3 *Evaluation Phase*

The evaluation phase is carried out by every peer P_i individually. No communication whatsoever is necessary. The root aggregate a_R confirmed by P_i and the majority of other requested peers provides the basis to compute the aggregation outcome. For the latter, a given deterministic algorithm is employed, so that all peers with the identical root aggregate a_R compute the identical aggregation outcome.

In the case of an election, the algorithm is imposed by the voting system, cf. Section 2.3. The computation of the outcome for the example of the FPTP voting system is detailed in Section 4.2.3.

5.4 PROTOCOL PROPERTIES AND IDENTIFIED ISSUES

The protocol properties of ADVOKAT are determined to a large extend by the model of a distributed protocol. Hence, its properties are to some extend similar to those presented in Section 4.5 of BitBallot. Both protocols are designed for largely autonomous operating peers with only few assumptions on synchronisation and connectivity, respectively peer presence, in order to match the network environments such as the internet. In this work, a mathematical rigorous specification of the actual communication model and the capacity of adversaries has been omitted for two reasons. First, the mere size and complexity of the internet renders the task difficult to provide a realistic and sufficiently precise model, which is a requirement for relevant security proofs of practical benefit. Second, at this early stage of the protocol, it is subject to continuous modifications and extensions. In consequence, the focus of this work has not been on security proofs. Further research is essential to deepen the understanding of the protocol.

For these reasons, the security-related and system-wide properties are discussed hereafter only on a high level.

5.4.1 *Security-Related Properties*

Common security properties of online voting protocols (cf. Section 2.1.2) are considered using the adversary model from Section 5.3.2. The protocol correctness includes *eligibility*, *integrity* and *accuracy* of the aggregation.

PROPERTY 6 (ELIGIBILITY): The aggregation algorithm ensures eligibility.

Proof (Sketch). The authority is trusted to provide only to eligible peers one authorisation token. The authorisation tokens can be verified by all peers using the public key of the authority. To get into the routing table of other peers, confirm aggregates, and request aggregates from other peers, one authorisation token and its corresponding secret key is required. Thus, non-eligible peers with no token cannot engage in the aggregation. □

The integrity and accuracy properties concern in aggregations the aggregates that have been cast already. ADVOKAT does not have a casting phase. In consequence, the properties must be interpreted and applied to the case of distributed protocols. Similar to Bitcoin, most properties are of probabilistic nature. Dishonest peers are assumed to be a minority among all present peers.

PROPERTY 7 (INTEGRITY AND ACCURACY): The aggregation algorithm ensures with high probability that aggregates obtained by sufficiently many peers cannot be altered.

Proof (Sketch). With no dishonest peers and peers linked to distinct leaf nodes, the accuracy follows from Property 1. In presence of dishonest peers, the integrity and accuracy are enforced by the honest peers. All computed aggregates must be signed. Early in the aggregation, manipulations of aggregates with small c, i. e. only few initial aggregates, can be recognised using proofs of deviations (cf. Section 5.3.2). In the unlikely event of a local majority of dishonest peers in one subtree despite the random and uniform distribution of all dishonest peers, manipulations may not be prevented. Distinct aggregate candidates with large c are chosen by a majority voting of their sources, that are with great probability honest by the majority and confirm the candidate computed with respect to the protocol.

Even if manipulations occur, their impact is bounded locally to the respective aggregate. Ancestors aggregates inherit manipulations. The Merkle hash tree ensures that child aggregates of computed and confirmed aggregates cannot be altered. □

PROPERTY 8 (CONFIDENTIALITY): The initial aggregates of honest peers are confidential.

Proof (Sketch). The protocol does not ensure secrecy of the initial aggregate due to the necessity to share it at least once over a pseudonymous channel. However, the access to the initial aggregate is limited to randomly determined peers that acquire mostly partial knowledge, so that confidentiality is ensured to a high degree. The pseudonymous channel between peers augments further the confidentiality. The DHT is ephemeral, distributes information evenly among peers, and vanishes when peers disconnect after the aggregation. Potential data breaches are therefore local and bounded in time. □

ADVOKAT does not provide *receipt-freeness*. Peers can deliver all aggregates from their initial aggregate up to the root aggregate confirmed by majority as a proof to a third party. Consequently, the protocol does not ensure *coercion-resistance* either.

Fairness, as presented in Section 2.1.2 and introduced in most works on voting protocols, requires that no peer gains information on intermediate aggregates before the end of the casting phase. With no casting phase, this property must be adapted to the case of distributed aggregation protocols. Less strict, fairness is given if peers have the same limited amount of information.

PROPERTY 9 (FAIRNESS): With great probability, every peer gains only information on few foreign aggregates before it cannot alter any longer its initial aggregate.

Proof (Sketch). The protocol requires that every peer pulls initial and intermediate aggregates from other peers. That means, peers learn initial aggregates at the very beginning and may postpone the computation of their initial aggregate. Eventually, every peer must provide its

confirmed initial aggregate and from then, changing the initial aggregate is a manipulation that can be detected. Deadlines are employed, so that peers confirmed to be present due to their signed requests cannot arbitrarily postpone their response to pull requests.

In the unlikely case of a non-uniform tree configuration with isolated peers with more empty sibling subtrees than in average, the fairness is diminished locally, because isolated peers may learn a larger intermediate aggregate than other peers. □

PROPERTY 10 (VERIFIABILITY): The aggregation phase is verifiable.

Proof (Sketch). Using requests, peers can determine with high probability which root aggregate has been confirmed by most peers and verify the chain of aggregate IDs that constitute a linked list of hashes from the root to the initial aggregate. □

The concept of implicit verification follows closely the verification based on on-spot observation of paper-based voting. In both cases, peers verify that their initial aggregate is counted as intended. Furthermore, a limited number of other initial aggregates are verified as well. While the limit in ADVOKAT is given by the tree configuration, it is in the case of paper-based voting the repartition to voting districts with voting offices run in parallel. ADVOKAT does not allow for a verification after the aggregation phase when most peers left the network. This is similar to paper-based voting if the safe storage of cast ballots is considered impossible. In fact, the immutable storage cannot be easily verified with no trusted authority.

5.4.2 *System-Wide Properties*

PROPERTY 11 (ROBUSTNESS AND NON-INTERRUPTIBILITY): The aggregation protocol is difficult to interrupt by a reasonable-sized coalition of dishonest peers.

Proof (Sketch). The aggregation step is entirely distributed to equipotent peers. With no weakest link, the influence of a reasonably-sized dishonest minority is locally limited. The redundancy of the aggregate computation increases exponentially while the aggregation approaches the root node and aggregates become more significant. □

The *protocol complexity* is mostly inherited from the properties of Kademlia, which have been studied (Cai and Devroye, 2013) and experimentally confirmed as part of BitTorrent.

MESSAGE COMPLEXITY For a network of n peers, a lookup requires with great probability $O(\log n)$ request-response cycles. Joining the network requires a limited number of lookups and is thus as well of order $O(\log n)$. With the consideration to estimate the number of empty

k-buckets from (Cai and Devroye, 2013), the average number of requests during the aggregation phase is found to be in $O(\log n)$. The global message complexity for the network of n peers is $O(n \log n)$, which is optimal for computations of any multiparty function in an accurate way with high probability in presence of a constant fraction of colluding dishonest peers (Gambs et al., 2011).

MEMORY COMPLEXITY The memory required to store non-empty k-buckets is in $O(\log n)$. Further, the aggregation algorithm requires to store $O(\log n)$ received aggregate containers for non-empty sibling subtrees and perhaps a limited number of alternatives in case of failing confirmations. Hence, for a constant size of aggregate containers in memory, the total memory complexity is again $O(\log n)$.

TIME COMPLEXITY Intermediate aggregates for ancestor nodes are computed in sequence. For a constant computation time of each aggregation operation and with an upper limit to request and confirm aggregates, the time complexity is $O(\log n)$.

5.4.3 *Summary*

The main issues of the distributed aggregation with BitBallot identified in Section 4.5.3 concern foremost its scalability and its vulnerability in face of large portions of absent peers and few colluding dishonest peers, that share credentials.

Those issues are resolved by ADVOKAT–foremost by a compromise in favour of correctness and in disfavour of confidentiality. BitBallot's pull principle relies on a less efficient pull of random child aggregates. If timing attacks and overviewing of all channels of a given target peer are neglected, BitBallot provides stronger confidentiality, because peers cannot establish with certainty the link between initial aggregates and their source. The probability to guess such a link correctly depends on BitBallot's tree arity k.

ADVOKAT is based on a sparse tree with arity 2. For every parent aggregate only one other, foreign child aggregate needs to be acquired that can be identified as such. During the aggregation phase, every peer P_i has to respond to a pull request with its initial aggregate a_i at least once, which allows the receiver to establish with certainty the link between the initial aggregate a_i and the public key pk_i of P_i or, if no anonymous channels are employed, P_i.

In contrast, ADVOKAT offers an optimal message complexity and a reasonable memory complexity for equipotent honest peers, that allow for large-scale aggregations. Non-colluding dishonest peers are defeated efficiently by the introduction of confirmations. Colluding dishonest peers cannot employ mutually secrets to manipulate the majority voting among aggregate candidates. Further, ADVOKAT deals el-

egantly with absent peers. The authority does not learn the leaf nodes of peers, which contributes to the security of the protocol.

5.5 IMPLEMENTATION AND SIMULATION

The aggregation phase of the protocol has been simulated on the basis of `kad`, an implementation of Kademlia[4] written in JavaScript. For this reason, the simulation has been written in JavaScript, too. This has the added benefit that the same code base can be reused for demonstrators built upon HTML5 technologies that are supported by a wide range of devices. Further, JavaScript has an event loop built-in, which is leveraged to handle the asynchronous message exchange among peers.

5.5.1 *Implementation Details*

`kad` comes with a set of extensions. `kad-spartacus`[5] allows to mitigate Sybil attacks by proofs of KID ownership based on cryptographic signatures (cf. Section 3.2.1.1). For each peer P_i, key pairs (pk_i, sk_i) are generated using elliptic-curves cryptography. The KID x_i of each peer P_i is derived by hashing pk_i first with SHA-256 and the result again with RIPEMD-160. It is assumed that the use of pk_i instead of the authorisation token t_i leads to an equally random distribution of KIDs, so that the generation of t_i can be omitted in the simulation. A simulation parameter allows to vary the deterministic generation of key pairs and consequently the KIDs, so that different tree configurations can be tested.

The simulation does not consider absent or dishonest peers. Absent peer do not have an impact on the aggregation phase simulated here and can be omitted without loss of generality. Non-colluding dishonest peers would fail to gather the signatures from other peers required to confirm manipulated aggregates and can be omitted for the same reason. Though, colluding dishonest peers may have the chance to confirm manipulated aggregates in the unlikely event of a local majority of dishonest peers. The development of collusion strategies for successful manipulations has been out of scope for this simulation.

To bootstrap the simulation, all n peers are instantiated and every P_i connects to the Kademlia network using an initial contact P_{i-1}. According to the Kademlia protocol, peers update their routing table using lookup requests. The simulation neglects churn, i. e. peers do not join or leave during the aggregation, so that the routing table does not change throughout the aggregation phase. Once all routing tables are complete, peers start the aggregation step like detailed in Section 5.3.2.

4 https://kadtools.github.io, v1.6.2 released on November 29, 2016
5 https://github.com/kadtools/kad-spartacus, v0.0.6 released on January 24, 2016

If a peer receives a request for an intermediate aggregate that has not yet been computed, the response is delayed. The aggregation steps in the simulation use neither parallel requests nor time-outs for requests.

5.5.2 *Evaluation*

In the following, a subset of properties discussed at a high level in Section 5.4 are evaluated. The convergence is measured by comparing the computed root aggregate ID of all peers. In the simulation, all peers compute the same value for all tested simulation parameters, i. e. convergence is given.

Next, the issue of confidentiality is considered. Both the degree of leakage of initial aggregates, and the concentration of knowledge on initial aggregates is measured. For that purpose, it is assumed that all initial aggregates are distinct.

The *leaked information* L_i of a peer P_i is defined to be the sum of the inverse of the counters of all containers that P_i used to respond to pull requests. $1/c$ denotes the probability to correctly link the contained initial aggregate of P_i to pk_i and in the case of non-anonymous communication channels to the address of P_i. The leaked information L_i is at least 1, because in a non-trivial aggregation with $n > 1$, P_i must respond at least once with its initial aggregate container with $c = 1$. In a perfectly balanced tree with $n = 2^B$ peers, L_i is strictly smaller than 2:

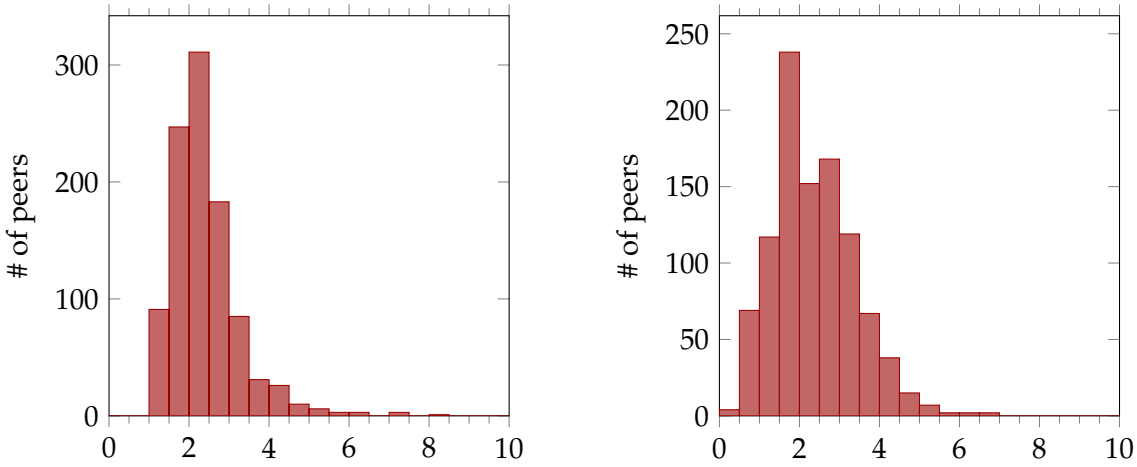
$$L_i = \sum_{n=0}^{B-1} \left(\frac{1}{2}\right)^n < 2$$

Conversely, the *received information* R_i of P_i is defined as the sum of $1/c$ of all containers that P_i receives as responses to its pull requests. In a perfectly balanced tree, $R_i = L_i$.

Further, relative measures $l_i = L_i/(n-1)$ and $r_i = R_i/(n-1)$ are introduced that are normed by the worst case, i. e. initial aggregates of all other $n-1$ peers are leaked/received. Figure 5.6 shows the distribution of L_i and R_i for a simulation run with $n = 1000$ peers. The simulation has been repeated with different tree configurations without notable changes. In the examined case, the relative leak to the network is $l_i = 0.24(9)\,\%$ which is approximately $2/(1000-1)$, the optimal value. The relative received information $r_i = 0.24(10)\,\%$ is the same with a slightly higher standard derivation.

The worst case is given by the least balanced tree configuration in which $|\mathbb{S}(x_i, d)| = 1$ for all $d \in \{B, \dots, 1\}$. That means for the given P_i, every sibling subtree contains exactly one other peer. Here, P_i learns with every pull request one initial aggregate with certainty. However, such a tree allows for only $B + 1$ peers and every additional peer decreases L_i.

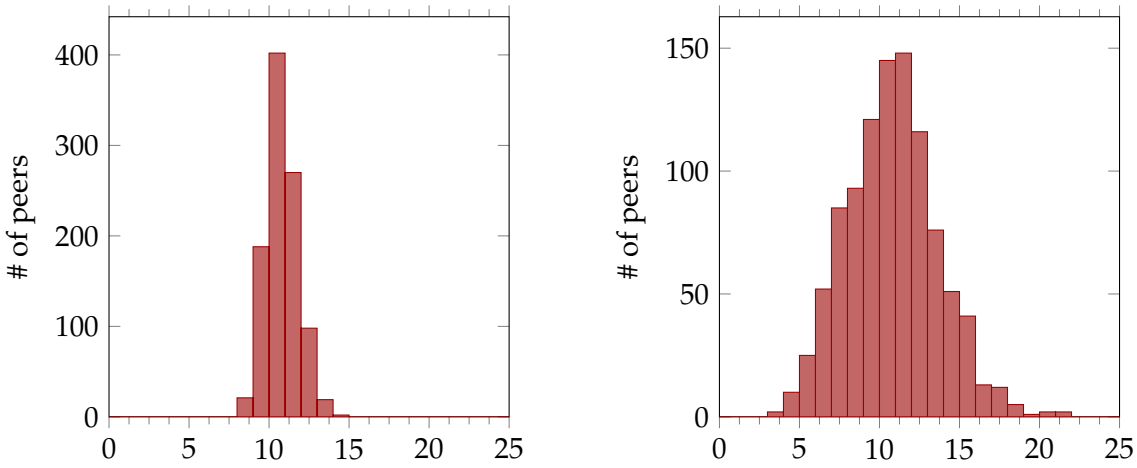
Moreover, the load on peers measured by the number of received and given responses has been examined. The histograms in Figure 5.7



(a) Histogram of leaked information L_i.

(b) Histogram of received information R_i.

Figure 5.6: Leaked and received information in ADVOKAT. In a simulation with $n = 1000$, peers leak (a), respectively receive (b), information on initial aggregates depending on the global distribution of peers on the binary Kademlia tree. L_i peaks close to the theoretical value 2 of an optimally balanced tree. Only few peers leak significantly more. While the mean for R_i is the same, the distribution is slightly different.



(a) Histogram of # of received responses.

(b) Histogram of # of given responses.

Figure 5.7: Received and sent requests in ADVOKAT. In a simulation with $n = 1000$, the number of given (b) and received (a) responses has been recorded for every peer. While the distribution of received responses is very sharp, the distribution for given responses is twice as broad. In the Kademlia routing tables, some peers are more often represented than others.

indicates that no peer receives significantly more load than others—a property that has been shown for Kademlia before.

Eventually, the average number of requests per peer, simulated with different numbers of peers n up to $n = 1000$, confirmed the theoretical message complexity of $O(\log n)$ shown in Section 5.4.2.

5.6 SUMMARY

In this chapter, the distributed protocol ADVOKAT has been introduced and its properties have been discussed. It is the outcome after many iterations of modifications to BitBallot. Inspired by the simplicity of paper-based voting, Bitcoin and Kademlia, ADVOKAT has been developed with an aim to keep the protocol simple while still allowing for an interesting set of properties. Compared to classical online voting, only basic cryptographic building blocks are employed, notably cryptographic hashing functions and asymmetric cryptography for encryption, signatures, and blind signatures. However, the complexity of ADVOKAT lies in the communication among many peers. Here, especially the complexity of the confirmation and recursive error correction mechanism stands out. For aggregations with no dishonest peers, no confirmation requests are required and the error correction mechanism can be omitted, which reduces further the complexity of the protocol.

It is indeed the recursive error correction mechanism, that introduces many protocol parameters that require tuning to the specific adversary model.

- The time-out for confirmation requests to confirm an aggregate candidate must be defined.
- The constraints on the sample used to estimate if a majority has computed a distinct aggregate candidate during the confirmation must be defined. A simple option is to define a minimum sample size. More complex options may consider the relative difference of the number of signatures of the two candidates with the most signatures.
- To ensure that invalid signatures from dishonest peers with proof of deviation are recognised as such, lookup requests in the DHT are used. For efficiency reasons, the lookup should be carried out only for a subset of the acquired signatures, that must be defined.
- A deadline must be defined after whose expiration no newly discovered peers should be requested for initial aggregates any more.
- A value for the confidentiality parameter c^*_{min} must be defined.
- An upper limit of the number of requests appended proofs of deviations must be defined.

- The error correction algorithm must allow the confirmation of more complete aggregates with larger c. However, if the conditions to start error correction are easily met, dishonest peers may slow down the aggregation by producing manipulated aggregates with arbitrary high value of c. Hence the error correction should be started only if a given number of signatures have been obtained already or otherwise constrained.

- A number of potential termination conditions are proposed and a definitive condition must be assembled. While in absence of dishonest peers, a timeout is sufficient, a different approach may be acquired to take into account potential forks that lead to diverging root aggregates.

A detailed analysis of the influence of the parameters and an in-depth study of the protocol properties are needed to bring the protocol onto solid grounds and allow its use in confidentiality-critical applications.

A number of ideas to further iterate the protocol are left as open research topics. There seems to be potential to improve the security of the aggregate confirmation in case of tree configuration edge cases, such as a very unbalanced tree. For this, the required signatures may depend on the actual tree configuration that is transparent for sibling subtrees with less than k peers.

The presented aggregation protocol relies on peers that are responsive throughout the aggregation phase. Similar to postal voting, a protocol extension may allow a small subset to store their initial aggregate in the DHT along with their contact information, so that it is disseminated to a random subset of other peers as the routing tables are populated. Like in postal voting, peers do not participate in the aggregation phase and verification phase.

Another topic for future improvements may be the revocation of signatures of aggregates that have been updated due to error correction. This could be achieved by the introduction of a new version parameter v that is provided along every aggregate signature, similar to the aggregate ID h and the depth d, and is required to validate the signature $\sigma_i(h, c, d, v)$. Each peer P_i increments the version parameter v from zero on separately for every subtree with depth d. If other peers obtain another signature of P_i with equal d and different v, previous signatures with lower v can be discarded. In consequence, the majority vote preceding an error correction better reflects the recent intermediate results.

6 DISTRIBUTED ONLINE VOTING

If more people were actively engaged in advocating their positions I think we'd have a better society.

— Jeb Bush (American businessman and politician)

This chapter is dedicated to a distributed online voting protocol that takes advantage of ADVOKAT as a middleware. The aim of the protocol is foremost to explore the feasibility of its implementation given today's web standards and a familiarisation with the potential and limits of ADVOKAT. The application is meant for demonstration purposes and may serve as a blueprint for other applications that rely on confidential aggregations.

After a short introduction, the protocol is described in Section 6.2. A JavaScript implementation presented in Section 6.3 allows to carry out votings in the browser. The chapter concludes with a summary.

6.1 INTRODUCTION

Major votings in 2016, e.g. the Brexit referendum or the US presidential elections, demonstrated the importance of a high voter turnout for the legitimacy of the outcome, especially in the case of tight outcomes.

While online voting is generally considered with much hesitation, advances in technology are eroding the security of paper-based voting as detailed in Section 2.4.5. Coercion-freeness, in the past a major argument for on-site ballot casting, is at stake due to omnipresent smart phone cameras. Exit polls on social media allow voters casting their ballot very late a more informed choice, which harms the fairness. More and more voters use early postal voting sacrificing thus their means of verification and the potential to change their vote last minute. Meanwhile, many online voting protocols seek to achieve those security properties that paper-based voting with optional postal voting can ensure less and less.

In this situation, distributed online voting offers a promising perspective. It does not assume a trusted authority and the integrity of the voting is enforced by the voters themselves. Without a weakest link such as a trusted authority, votings are difficult to interrupt. The damage in case of security breaches is locally bounded thanks to the distribution of data. Like in all online voting protocols, voting becomes more convenient, especially as ballots can be cast remotely. Though, it must be acknowledged that still trust in technology or expert knowledge is assumed.

6.2 PROTOCOL DESCRIPTION

The aggregation protocol described in Chapter 5 is used as a middleware in the distributed online voting protocol denoted ADVOKAT-Vote which allows to showcase the aggregation algorithm in a classroom or lab environment. The protocol is not designed for elections of officials in contexts of high stake.

The assumed model consists of peers P_i, here also called voters, and one sponsor of the vote denoted S. The sponsor trusts and has access to an API that is used to setup the voting and prepare the aggregation. The API implements foremost the authority A required by ADVOKAT and is threefold:

1. The registry A_R is responsible to provide one authorisation token t_i for every eligible voter P_i.
2. For practical reasons, the registry A_R employs a standardised central authentication service (CAS) denoted A_{CAS} to identify the voters P_i. The CAS may be implemented by e. g. a university or a social network.
3. At least one tracker T is required by ADVOKAT for the initial voter discovery.

Is the dilemma of the distributed definition of the electorate the chicken-egg problem of the social choice theory?

An open research question common to many online voting protocols is the registration of voters with no trusted authority. The eligibility can only be assessed after the electorate is defined with consensus. However, it is unclear how such a consensus could be reached with no authority and no consensus on an electorate to survey.

Moreover, ADVOKAT does not provide a mean to prevent ballot stuffing, i. e. the illegitimate creation of authorisation tokens by the authority for non-eligible, non-existing or already authorised voters. Hence, for ADVOKAT-Vote, it is assumed that the sponsor S is trusted to choose the electorate and has chosen a trusted API that generates on request for each eligible voter one authorisation token and not more.

Without loss of generality, the voting protocol employs approval voting m out of n. Other voting systems covered in Section 2.3 are possible. One voting includes one or multiple voting questions with each a predefined set of potential answers. For each question, the maximum number of approvable answers m is given. In the limit with $m = 1$, the voting protocol provides a simple 1 out of n FPTP voting.

The description of the voting is stored in a *voting description file* denoted f and contains the voting questions, their answers, and configuration options required by the aggregation protocol. It is similar to the torrent file of the BitTorrent protocol (Section 3.2.2.1). In order to distinguish different votings from one another, voting IDs are employed that are the hash $\eta(f)$ of the voting description file.

In addition to the communication channels required by ADVOKAT, any channel suited to share f among the electorate is required. Further, a confidential channel is required between each voter and the CAS.

The setup of the voting by the sponsor S and the preparation by each P_i are given in Figure 6.1 and detailed hereafter. The aggregation phase and evaluation is carried out as described previously in Section 5.3.

6.2.1 *Setup Phase*

The sponsor carries out the setup of the voting as follows.

1. The voting sponsor acquires from the registry A_R on the authenticated, tamper-resistant communication channel a new public key pk_A for a voting and the CAS API endpoints of the A_{CAS} that A_R uses to verify identity tokens.
2. S creates the voting description file f containing the voting questions and with each a predefined set of potential answers plus the maximum number of approvable answers m for each question. Further, the configuration of the aggregation (cf. Section 5.6) not predetermined elsewhere is included, most importantly the deadline of the preparation phase and the termination condition. The description file must be designed in a way that does not allow for identical votings with distinct description file. In consequence, the mutation of the description file leads to a different voting ID $\eta(f)$.
3. S sends the definition of the electorate and the deadline of the preparation phase to A_R and awaits the acknowledgement.
4. Only afterwards, S notifies the electorate about the voting and shares f among them. The voting description may be public, so that arbitrary parties may publish f.

6.2.2 *Preparation Phase*

The preparation phase of each peer P_i starts with the reception of the voting description file f, that it receives using the channel dedicated for voting notifications.

1. According to ADVOKAT, P_i generates a key pair (pk_i, sk_i).
2. Using an authenticated, tamper-resistant channel, P_i provides to the A_{CAS} defined in f a proof of identity and the voting ID $\eta(f)$. It receives in return its identity token IT bound to $\eta(f)$.
3. P_i sends its token IT, $\eta(f)$ and the blinded public key b_i to A_R. The registry verifies the token IT which may require a request to A_{CAS},

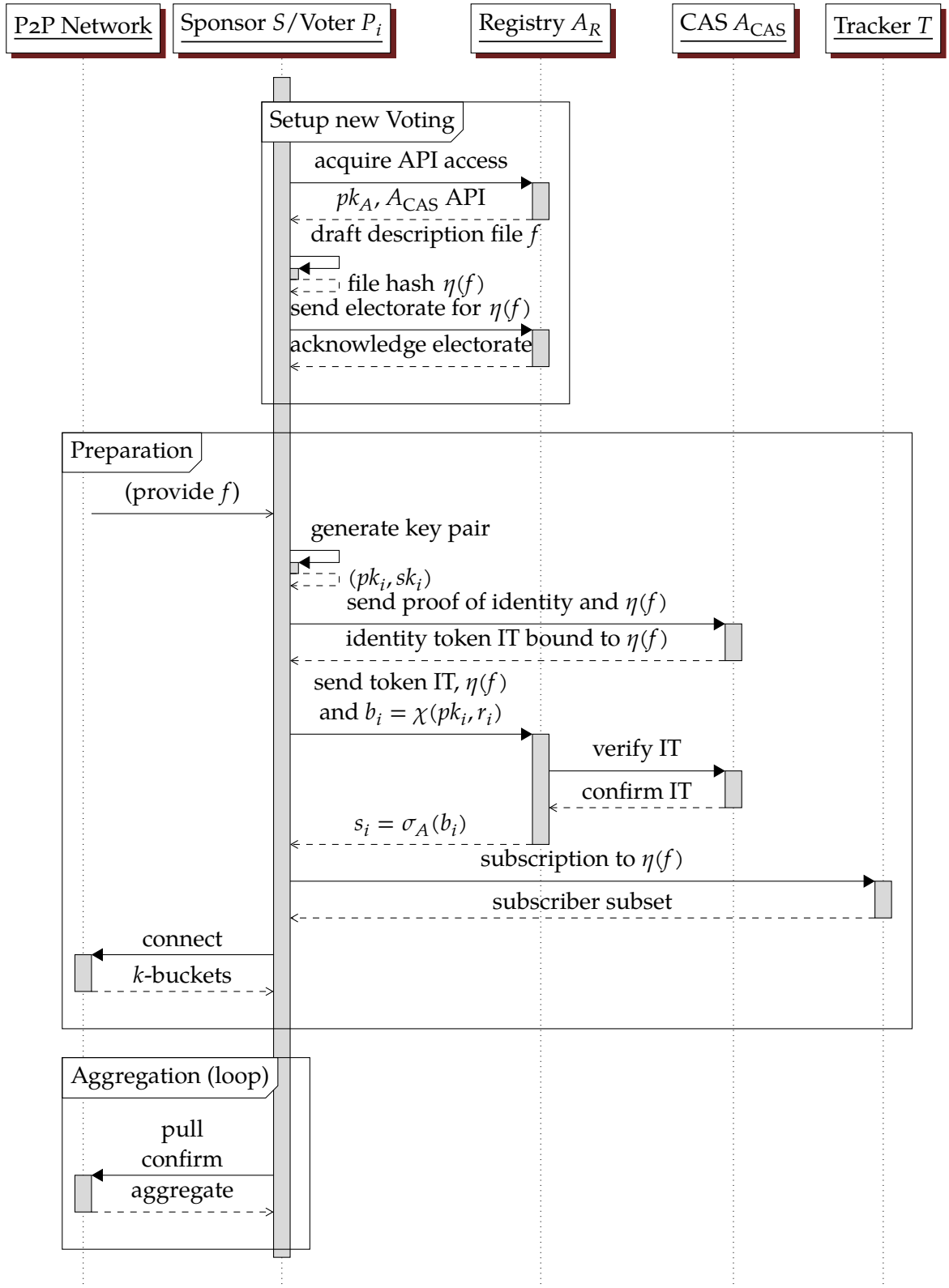


Figure 6.1: ADVOKAT-Vote sequence diagram. The steps of the sponsor S to setup a new voting and the consecutive preparation of every voter P_i is given. The authorities A_R and A_{CAS}, and the tracker T are not required any longer afterwards.

ensures it is bound to $\eta(f)$ and if found valid and eligible, returns to P_i the signature s_i on b_i. No further b_i is signed for P_i.

4. P_i unblinds s_i to obtain its authorisation token t_i. Further, it subscribes to the voting $\eta(f)$ at the tracker T and receives in response a subset of other subscribers of $\eta(f)$.

5. P_i joins the voting network and populates its routing tables according to ADVOKAT.

The deadline of the preparation phase marks the end of this phase. The registry A_R and the CAS A_{CAS} discontinue their service for $\eta(f)$.

6.2.3 *Verification of the Registration*

As stated previously, both authorities must be trusted to register only eligible voters. While a malicious deviation cannot be prevented, the operation of the authority can be verified and deviations may be detected in some cases.

For this, the protocol can be extended by a registration verification phase between preparation and aggregation phase. A_R must provide the signatures s_i for every P_i using a broadcasting channel, e. g. BitTorrent, so that peers can verify the total number of registered voters and verify that their own signature is mentioned. In case of detected deviations, a proof of deviation is shared among all voters and the voting is interrupted and may be setup again with a different choice of A.

6.3 IMPLEMENTATION

A basic demonstrator denoted ADVOKAT-Vote has been implemented to test the protocol in a browser environment. It consists of two components. First, a backend written in JavaScript provides the API for the registry A_R and the tracker T according to Section 6.2 and Figure 6.1. The backend interacts with the CAS A_{CAS} provided by a third party. For the demonstrator, the Google+ API of Google is employed.

Second, a JavaScript-based HTML5 frontend application allows a sponsor S to setup votings and voters P_i to prepare their ballot and carry out the aggregation. The frontend is entirely static and is downloaded from a webserver. However, it may also be distributed as a native application using mobile application frameworks such as Apache Cordova. The frontend shares the same code base of ADVOKAT that has been used in Section 5.5 for the simulation. The frontend library AngularJS, which has been used for the Bitballot implementation recalled in Section 4.4, is used for the control logic and the library Angu-

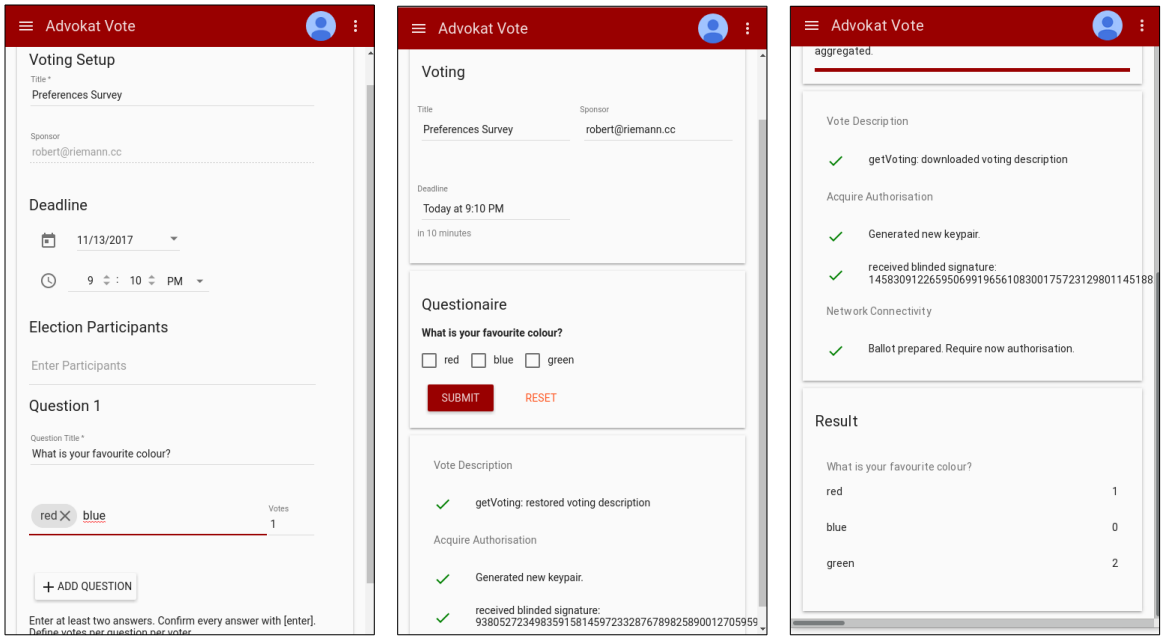
larJS Material[1] for the web interface. Unlike Bitballot's implementation, WebRTC[2] is used for P2P message passing directly between the voters' browsers. In order to establish a connection between two browsers, the exchange of signaling meta data is required. In the case of the demonstrator, the backend realises the exchange using WebSockets. As of the 8 November 2017 draft of WebRTC 1.0, signaling data cannot be reused to open new connections to other browsers. In consequence, the communication depends to a large extend on the backend which contradicts the idea of a distributed protocol. Yet, one can imagine improvements to WebRTC to resolve this limitation.

Due to the encountered incompatibilities of employed third-party cryptography components, the demonstrator does not issue, blind, unblind or verify signatures for messages or aggregate containers. The W3C recommendation of a browser-provided *Web Cryptography API* may improve this situation once it is adopted and fully implemented by browser vendors. Further, confirmation requests are not covered yet, so that in the case of protocol deviations distinct results may occur. While the registry A_R ensures that each Google+ account is registered at most once, the registration is not restricted to the electorate defined in the description file f of a voting to ease testing. An effective restriction would require to provide A_R with f as indicated previously.

Besides these adjustments, ADVOKAT-Vote follows the schema depicted in Figure 6.1. The sponsor S and voters P_i use the Google+ login to acquire a proof of their mail address from Google. The setup of a voting is shown in Figure 6.2a. The sponsor defines a title, a deadline to join the P2P network, the electorate and one or more questions with each two or more options. The maximum number of options to choose can be defined for each question, which allows to carry out basic FPTP, approval, and ranged votings (cf. Section 2.3). The form is then stored together with the previously downloaded pk_A and the link to the A_{CAS} API as a JSON-formatted description file f. This file must be provided to all voting participants. The demonstrator uses the backend tracker to distribute next to P2P network contacts also the description file f. Then, the sponsor shares a link containing $\eta(f)$ with the voters. After the download of the description file f from the tracker, the voters acquire a proof of their mail address directly from Google, and provide it together with their public key to the API endpoint of A_{CAS} indicated in the description file f. In return, they receive their identity token that is used to compute their KID. Subsequently, the voters register at the tracker and get the contact information of a random other voter, which is used to connect to the network. The connection to the P2P network

1 AngularJS Material (`https://material.angularjs.org`) is a JavaScript-based open-source graphics library for AngularJS web applications that implements the design language *Material Design* known from the Android OS.

2 *Web Real-Time Communication* (WebRTC, `https://webrtc.org`) is a W3C draft initially put forward by Google to allow for P2P connections among internet browsers using a simple JavaScript API.



(a) Setup Phase (b) Preparation Phase (c) Evaluation Phase

Figure 6.2: Screenshot of the frontend application ADVOKAT-Vote. The JavaScript-based frontend for the browser allows sponsors to create new votings (a). Given the link to access the voting, voters can prepare their ballot (b) and carry out the aggregation. After the aggregation phase, the voting outcome is displayed (c).

is established as soon as the voter has answered all questions using the form depicted in Figure 6.2b and created its initial aggregate. Once the deadline to join the network has expired, the aggregation phase starts. As soon as the root aggregate is computed, the voting outcome is displayed as shown in Figure 6.2c.

At this stage, tests have been run with few voters. The code of the implementation is licensed under GPL v3.0. The frontend is available at `https://gitlab.inria.fr/riemann/advokat-vote` and the backend at `https://gitlab.inria.fr/riemann/advokat-vote-backend`.

6.4 SUMMARY

The implemented applications in this thesis are based on JavaScript for the sake of rapid prototyping, wide platform coverage and ease of deployment to browsers. It must be noted though that secure delivery of JavaScript to browsers, and sandboxing is considered particularly difficult (Ptacek, 2011). Furthermore, the robustness of the implementations based on WebRTC are diminished, because the current version of its browser-to-browser communication protocol WebRTC relies on a signaling server to establish every new channel among peers.

The experience of implementing ADVOKAT-Vote using state of the art web technologies has raised the awareness of a number of pitfalls for academic use cases. Within the duration of this thesis, AngularJS and the Kademlia library `kad` have received major version releases with entirely new APIs. Many dependency libraries seem to follow the development philosophy *release early, release often*. A compromise between a stable code base and integrating bug fixes is difficult to find. The frontend of ADVOKAT-vote relies on about 130 third party software packages in sometimes more than one version. In consequence, much caution is needed to select software frameworks that provide at best extensive documentation, an active community and long-term support.

7 DISTRIBUTED ONLINE LOTTERY

Any one who considers arithmetical methods of produing random digits is, of course, in a state of sin.

— John von Neumann (Hungarian-American mathematician)

The application scope of confidential aggregation algorithms outlined in Section 2.6 exceeds the case of voting applications. To illustrate the flexibility of ADVOKAT, one case has been studied in particular. The contribution is a novel protocol for verifiable large-scale online lotteries with no trusted authority to carry out the random process, for which concepts originating from online voting are employed (Riemann and Grumbach, 2017c).

In the following section, the problem is motivated and introduced. Due to the small overlap of the scientific production on online voting covered in Chapter 3 and online lottery, a review of the current state of art on online lottery is provided briefly in Section 7.2. Then, a centralised online lottery protocol is recalled in Section 7.3 in greater detail as it is the basis for the distributed online lottery protocol based on ADVOKAT, which is introduced in Section 7.4. It follows a discussion of its properties in Section 7.5 and its implementation details in Section 7.6. The chapter concludes with a summary in Section 7.7.

7.1 INTRODUCTION

Lottery is a multi-billion dollar industry (Isidore, 2015). In general, players buy lottery tickets from an authority. Using a random process, such as the drawing of lots, the winning tickets are determined and the corresponding ticket owners receive a reward.

In some lotteries, the reward may be considerable, and so is the incentive to cheat. The potential of fraud gained attention due to the *Hot Lotto* fraud scandal. In 2015, the former security director of the Multi-State Lottery Association in the US was convicted of rigging a 14.3 million USD drawing by the unauthorised deployment of a self-destructing malware manipulating the random process (Rodgers, 2015).

In order to ensure fair play and ultimately the trust of players, lotteries are subject to strict legal regulations and employ a technical procedure, the *lottery protocol*, to prevent fraud and convince players of the correctness. Ideally, players should not be required to trust the authority. Verifiable lottery protocols provide therefore evidence of the correctness of the random process.

In a simple paper-based lottery protocol, tickets are randomly drawn under public supervision of all players from an urn with all sold tickets to determine the winners. Afterwards, all tickets left over in the urn are also drawn to confirm their presence and convince the losers of the correctness. Without public supervision, the random process can be repeated until by chance a predefined result occurred. Further, the process can be replaced entirely by a deterministic process.

In practice, the public supervision limits the number of lottery players and is further very inconvenient, because players are required to respect the time and locality of the drawing procedure. With the advent of public broadcasting channels, first newspapers, then radio and television broadcasting, protocols were employed that replaced the public supervision by a public announcement. Only few players and notaries verify here the correctness of the random process. In consequence, the majority of non-present players are required to trust the few present individuals for the sake of scalability. With the increasing availability of phones and later the Internet, protocols have been adapted to allow also the remote purchase of lottery tickets, e. g. from home or a retail store.

The technical evolution lead to a gradual change of how people play lottery, but in many cases, the drawing procedure has not been adapted and resembles more a legacy that prevails for nostalgic reasons than to provide security. In the simple paper-based lottery protocol, the chain of custody establishes trust. All operations may be inspected by eyesight.

This technique cannot be directly adapted for an online lottery. Thus, most verifiable online lottery protocols (Goldschlag and Stubblebine, 1998; Zhou and Tan, 2001; Chow et al., 2005) rely on a concept based on two elements. All players can actively contribute to the random process. Nobody can compute the random process result or its estimation as long as it is possible to contribute or buy new tickets. The latter is required to prevent educated contributions to circumvent the uniform distribution of the random process. It is the lottery protocol that must ensure the order in time of the contribution and the actual determination of winners.

A protocol consisting of equipotent players contributing each to the randomness of the publicly verifiable random process is promising for its similarity with the simple paper-based lottery protocol. Again, all players participate in the execution and supervision of the random process. The feasibility to construct such protocols with no trusted third parties has been demonstrated by Bitcoin (cf. Section 3.2.3.1). Lottery protocols based on Bitcoin have been already considered (Pierrot and Wesolowski, 2016), cf. Section 7.2.

Although different, the security requirements of lotteries share common concerns with those of voting systems. Both lottery and voting protocols have to assure trust in an environment of mutual distrust

among players, respectively voters, and the potentially biased authorities. The literature on voting protocols and online voting protocols is extensive and comprises flexible protocols that may be adapted to different voting systems beyond the general case of majority voting. Of particular interest for a lottery are online voting protocols that allow for a random choice. Already the common paper-based voting for general elections provides a solution to improve the scalability of the simple paper-based lottery protocol: the introduction of multiple offices that run in parallel. To highlight further the similarities, the security-related voting protocol properties presented in Section 2.1.2 are adopted for lottery applications as follows (in no particular order):

a) *Correctness of the random process*. All numbers are equally likely to win. Nobody can predict the random process better than guessing.

b) *Verifiability of the random process*. Players can be convinced that the random process has not been manipulated.

c) *Privacy of the player*. Players do not learn the identity of other players to prevent blackmailing or begging.

d) *Eligibility of the ticket*. Tickets cannot be forged. Especially, it is impossible to create a winning ticket after the outcome of the random process is known.

e) *Confidentiality of the number*. Numbers are confidential to ensure fairness. Tickets of other players cannot be copied to reduce their potential reward.

f) *Completeness of the reward*. Players can verify the number of sold tickets that may determine the reward.

7.2 RELATED WORK

Different protocols have been proposed that allow players to contribute to the publicly verifiable random process and take measures to prevent early estimations of the result while it is still possible to contribute.

A trivial solution in the context of secure parameters for cryptography is recalled in (Lenstra and Wesolowski, 2015). In a first round, all players choose secretly a number and publish on a public broadcasting channel a commitment on their number, e.g. using a hash. In a second round, all secret numbers are revealed and verified using the commitment. Finally, all values are concatenated using the XOR operation to form the result. The protocol owes its correctness to the clear separation in two rounds of player's contributions. However, the authors stress that the protocol is neither robust nor scalable. Termination is not possible if one player does not reveal its secret number and for the

verification, all players have to run as many XOR operations and send as many messages as there are players.

Subsequently, a random process protocol with only one round has been proposed (Lenstra and Wesolowski, 2015). A delay between player contribution and winner identification by the authority is imposed, so that estimations would be available only after contributions are no longer allowed. Players or other third parties can engage before a deadline in the collection of arbitrary data, e.g. from social networks like Twitter, to generate a seed, that is essentially a random bit string. Right after the deadline, the authority publishes a commitment on an additional, secretly chosen seed. Both seeds provide the input for a *proof of work*. A proof of work is computationally expensive to generate and thus time-consuming. A delay is inevitable. However, due to its asymmetry, the proof allows for efficient verification. Once the proof is found, the winners are derived from it. The additional seed prevents dishonest players to predict the results for different potential last-minute contributions. One has to assume that the last honest contribution is made sufficiently late to prevent the same attack from the authority.

Chow's online lottery protocol (Chow et al., 2005) published a decade earlier relies on a technique called *delaying function* (Goldschlag and Stubblebine, 1998) that is similar to a proof of work, but is not asymmetric and does not provide efficient verification. The authority commits here on the concatenation of the players' commitments on their secretly chosen number and derives then the winners. Players can claim the reward by publishing the input data of their commitment. Similar to (Lenstra and Wesolowski, 2015), a late honest player commitment is assumed to prevent a prediction by the authority. Then, all security measures introduced above are provided. The protocol requires players to process the commitment of all other players and recompute the delaying function in order to verify the random process, which is impractical for large-scale lotteries.

Solutions for a scalable probabilistic verification of online lotteries (Y. Liu, Hu and H. Liu, 2007) or online voting (Markowitch and Dossogne, 2010) have been presented based on a concatenation/aggregation over a tree structure. In order to verify the result at the root of the tree, players or voters can repeat the computation of intermediate results for a predefined or random subset of all tree nodes. With increasing number of verified nodes, the probability of a manipulated result at the root node diminishes.

Other online lottery protocols introduce mutually distrusting, noncolluding authorities to allow for a *separation of powers*. In (Kuacharoen, 2012), a distinct auditor ensures secrecy and immutability of the player's tickets and prevents the lottery authority from adding illegitimately tickets. For this, blind signatures and public-key encryption are employed. The protocol does not cover the random process and its verification. Authorities are assumed not to collude.

In (Fouque, Poupard and Stern, 2001), the secrecy of online lottery and voting protocols is addressed at the same time. A mechanism based on homomorphic encryption, distributed key generation and threshold decryption is proposed. A set of mutually distrusting authorities have to cooperate to decrypt the result of the random process or the voting. A colluding set of dishonest authorities below the threshold cannot reveal prematurely the result, i.e. to add a winning ticket in the lottery case. Players or voters are entitled to trust that the set of dishonest, colluding authorities does not meet the threshold. Ideally, the power to decrypt would be shared among all players or voters. Practically, this is often not feasible due to scalability issues.

The potential of the Bitcoin blockchain (Nakamoto, 2008) for a distributed random process has been examined. However, it has been shown that the manipulation of presumably random bits is realistic even with limited computational capacity and financial resources (Pierrot and Wesolowski, 2016). An integration of the proof of work from (Lenstra and Wesolowski, 2015) and an alternative crypto-currency Ethereum (Wood, 2014) has been proposed[1] with no practical solution yet for a verification due to the limitation imposed by the blockchain.

7.3 CENTRALISED ONLINE LOTTERY PROTOCOL

The starting point for the proposed protocol is the centralised online lottery protocol of Chow et al. (2005), recalled hereafter with an alternative verification based on hash trees (Y. Liu, Hu and H. Liu, 2007).

The following presentation of Chow's protocol with hash trees is reduced to aspects required for the proposition in Section 7.4. The following notation is employed:

A	Authority (Dealer in (Chow et al., 2005))
P_i	Player, i-th out of n
n_i	Number in the set $\mathbb{L}$ chosen by player P_i
r_i	Random bit string of given length chosen by player P_i
$\eta(\cdot)$	Cryptographic hash function, e.g. SHA-3
$\eta_0(\cdot)$	Cryptographic hash function mapping any r_i to $\mathbb{L}$
$\sigma_A(\cdot)$	Authority's signature scheme

Chow's protocol implements a lottery in which every player P_i has to choose a number $n_i \in \mathbb{L}$ and send a commitment on it to the authority A. A aggregates all commitments to a value h. That means, every P_i contributes to h. The aggregate h is used as an input parameter for a *delaying function* (DF) preventing A from early result estimations. The outcome of DF is used to compute the winning number n_R with a *veri-*

1 http://www.quanta.im, https://kiboplatform.net (accessed 02/02/2017)

fiable random function and the secret key of A. Players do not have the secret key required to compute n_R, but can verify n_R using the public key of A.

During the *ticket purchase* phase, P_i acquires from A a personal sequence number s_i. P_i has to choose its number n_i and a random bit string r_i to compute its commitment ticket_i with bit string concatenation $\|$ and XOR operation $\veebar$. P_i sends ticket_i to A and receives in return the signature $\sigma_A(\text{ticket}_i)$ as a receipt.

$$\text{ticket}_i = s_i \| (n_i \veebar \eta_0(r_i)) \| \eta(n_i \| s_i \| r_i)$$

The DF cannot be evaluated before h depending on all commitments is given, which is ideally only after the purchase phase. In (Chow et al., 2005), the DF input parameter h is recursively computed from all n commitments with $h = \eta(\text{chain}_n)$ and $\text{chain}_i = \eta(\text{chain}_{i-1} \| \text{ticket}_i)$ with an empty initial chain chain_0. An alternative introduced in (Y. Liu, Hu and H. Liu, 2007) consists of a computation of h using a T-ary Merkle tree (Merkle, 1988) with ticket_i assigned to the leaf tree nodes. In both cases, all ticket_i are published to allow the verification of h by the players requiring memory and computational resources of respectively $O(n)$ and $O\left(\log_T(n)\right)$.

Once the authority has published the verifiable winning number n_R, the *reward claiming* phase begins in which players P_i with $n_i = n_R$ provide their sequence number s_i and their secret random value r_i to A via a secure channel. Upon verification of the commitment ticket_i by A, the reward is granted. P_j with $n_j \neq n_R$ may verify that their commitment ticket_j has been used to compute h and are assumed to have trust in the infeasibility of A to compute DF more than once between the latest honest ticket contribution and the publication of n_R.

7.4 DISTRIBUTED ONLINE LOTTERY PROTOCOL

With the distributed aggregation protocol ADVOKAT and elements of the centralised online lottery, an online lottery with distributed random process is assembled. It is run by an authority that handles the ticket purchase and carries out the distribution of the reward upon winner verification, but not the random process itself. The random process is distributed to all players using ADVOKAT.

The proposed protocol allows for a lottery with playing mode CL or LO (Kuacharoen, 2012):

CLASSIC LOTTERY (CL) Rewards are distributed with respect to a randomly ordered list of all players.

LOTTO (LO) Rewards are distributed based on the secret, prior choice of each player. The secret can be e. g. one or more numbers in $\mathbb{N}$.

The protocol has six phases of which *ticket purchase*, *reward claiming* and *winner verification* follow closely Chow's protocol (Chow et al., 2005). Its model provides for an authority A, a tracker and players P_i. For CL, $\eta_0(\cdot)$ is identical to $\eta(\cdot)$, so that the root aggregate a_R of the distributed aggregation is in the domain of the KIDs.

7.4.1 *Setup Phase*

1. A generates a key pair (pk_A, sk_A) and chooses a random bit string r_A.
2. A publishes the deadline of the ticket purchase, pk_A, $\eta(\cdot)$, $\eta_0(\cdot)$ and $\eta(\eta(r_A))$. Further, A specifies the duration of the aggregation epoch for each tree level.

7.4.2 *Preparation Phase (Ticket Purchase)*

1. P_i picks a random string r_i, and for LO its number $n_i \in \mathbb{N}$. It generates (pk_i, sk_i).
2. P_i sends pk_i to the authority and obtains in return a sequence number s_i and its authorisation token $t_i = \sigma_A(pk_i)$. Different than in previous chapters, no blinding scheme is employed during the token generation.
3. P_i computes $x_i = \eta(t_i)$ and connects to the Kademlia DHT using an already connected contact provided by the authority or a separate tracker.
4. P_i prepares its initial aggregate $a_i = \eta(\text{ticket}_i)$. For CL, $\text{ticket}_i = s_i \| r_i$ and for LO, $\text{ticket}_i = s_i \| (n_i \veebar \eta_0(r_i)) \| \eta(n_i \| s_i \| r_i)$, c.f. Section 7.3.

7.4.3 *Aggregation Phase (Distributed Random Process)*

1. After the ticket purchase deadline, A publishes the number of sold tickets n.
2. All P_i compute jointly the root aggregate a_R. The $\oplus$-operation is given by $a_i \oplus a_j = \eta(a_{ij})$ with $a_{ij} = a_i \| a_j$, if $a_i < a_j$, otherwise $a_{ij} = a_j \| a_i$. It is a commutative variant of the binary Merkle tree scheme proposed in (Y. Liu, Hu and H. Liu, 2007). The security parameter l defined in Section 5.3.2 may be increased to require more distinct signatures to confirm aggregates.
3. Proofs of protocol deviation in form of pairs of signatures are sent to A that can reveal the corresponding players and revoke their right to claim a reward. This is possible, because no blind signature scheme has been involved to obtain t_i.

7.4.4 *Evaluation Phase (Winner Identification)*

In order to identify the winner, A must determine the a_R of the majority.

1. A requests a_R of multiple random P_i until a majority of the sample meeting a predefined minimum confirmed one a_R.
2. A publishes a_R, r_A and the winning number $n_w = \eta_0(a_R) \veebar \eta_0(r_A)$.
3. For CL, A computes all $x_i = \eta(t_i)$, orders all P_i by their Kademlia XOR distance $d(n_w, x_i) = n_w \veebar x_i$ and players on a par by $n_w \veebar s_i$, and publishes as many ordered x_i as there are rewards. For LO, winners P_i have $n_i = n_w$.

Then, the winners must actively claim their reward.

1. The winner P_i sends all its confirmed aggregate containers to A to proof their participation. For LO, P_i must also provide its ticket$_i$ and (s_i, r_i).
2. Proofs are published for verification by other players.

7.4.5 *Verification Phase*

1. $\eta(\eta(r_A))$ is computed for comparison with the previously published value and n_w is verified.
2. Players verify that winner P_i took part in the aggregation by comparing its published containers with their computed containers.
3. For CL, player verify the order of the published winners and compare it to their own positioning. For LO, ticket$_i$ is reproduced for the given (s_i, r_i) and its hash must equal a_i found in the published confirmed aggregate containers.
4. If the rewards depend on the number of sold tickets n, n is compared to the counter c of the root aggregate container.

7.5 PROTOCOL PROPERTIES

The protocol is analysed with respect to the security properties introduced in Section 7.1 under the adversary model from ADVOKAT in Section 5.3.2 of an adversary D controlling a fraction $b < 0.5$ of dishonest players of n players in total. The performance of the protocol depends upon b, the security parameter l and the tree configuration, i. e. the distribution of honest and dishonest players over the tree. It is assumed that D and A collude.

7.5.1 *Most Likely Scenario*

Due to the uniform player distribution and for a reasonably sized b, D has most likely a dishonest majority only in subtrees with large depth $d > 1$ containing only a small number n' of players. l can be adjusted to detect container manipulations of subtrees with $n' \leq l$ using signatures. Most likely, all dishonest players have to provide a container with their signature to at least one honest player, which corresponds to a commitment to their ticket_i, before D can learn all containers for a given depth d.

1. The *correctness* of the random process and its implicit *verification* (Y. Liu, Hu and H. Liu, 2007) due to the distributed computation is with great probability ensured, because D cannot change or add tickets after a prediction becomes possible.

2. The *privacy* of players is ensured. Other players cannot learn the identity of each other from the exchanged messages.[2]

3. The authorisation token t_i ensures *eligibility*. A participation after the aggregation has started even with a valid t_i is unlikely, because honest players close in the tree deny belated players and do not confirm their containers.

4. The commitment scheme for LO provides *confidentiality*, because number n_i of P_i cannot be revealed without knowledge of the secret r_i (Chow et al., 2005).

5. The counter c of the root aggregate container allows to examine the *completeness* of the reward.

7.5.2 *Worst Case Scenario*

The distribution of honest and dishonest players is highly unbalanced. A majority of at least l dishonest players in a subtree $\widehat{\mathbb{S}}(x_e, d)$ for some d is assumed. Neither the majority nor the confirmation criterion prevent a manipulation with certainty. The local minority of honest players may be excluded from the aggregation unable to proof their participation. As the manipulation is bounded locally, correctness and eligibility are only locally violated.

If D has further in all other non-sibling subtrees $\widehat{\mathbb{S}}(\cdot, d)$ at least one dishonest player to provide the local aggregate container, D can compute with the secret r_A from A the winning number n_w while the container for $\widehat{\mathbb{S}}(x_e, d)$ is not yet known to honest players and may be altered to change n_w. A collusion with A and a proof of work is required to

2 The leak of the identity due to the communication channel, e.g. by the IP address, may be solved using privacy networks like Tor and is out of the scope of this thesis.

choose a particular n_w. Here, the correctness of the random process is undermined.

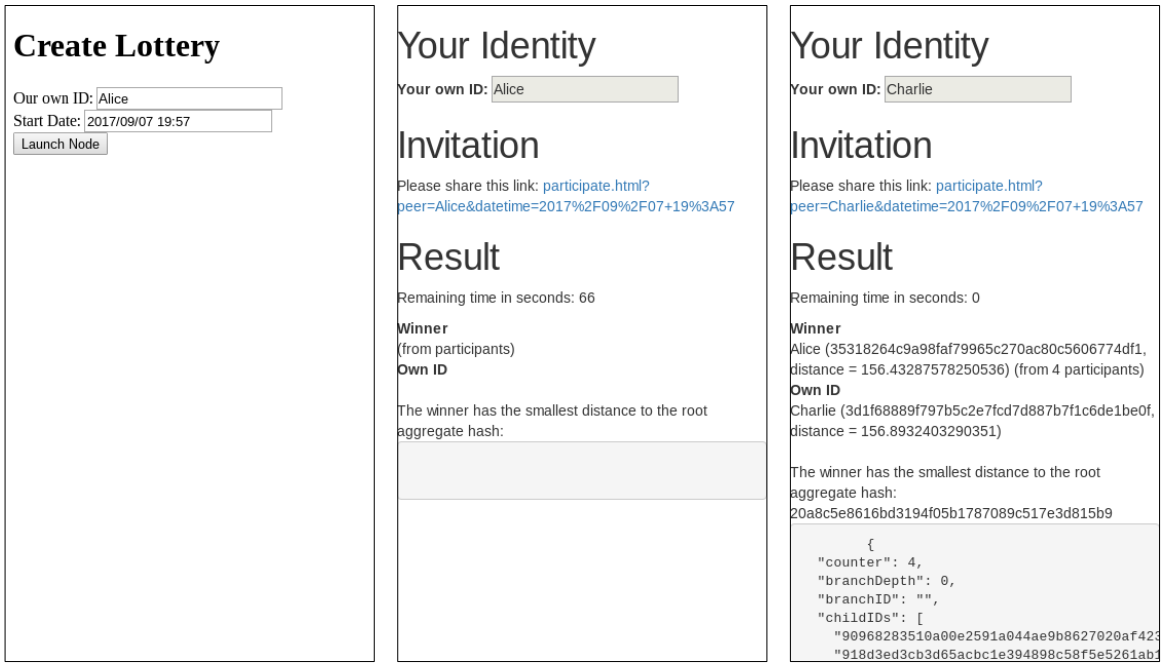
The distribution of n' honest or dishonest players to $\widehat{\mathbb{S}}(x_e, d)$ and its sibling subtree $\mathbb{S}(x_e, d)$ follows the Binomial distribution $B(n', p)$ with $p = 0.5$ and a variance of the ratio of players in $\widehat{\mathbb{S}}(x_e, d)$ of p^2/n'. As a result, the probability of a local dishonest majority decreases reciprocally in n'. n' decreases for increasing d, but for large d, it is unlikely to have a dishonest player in all non-sibling subtrees for the limited number of dishonest players. In consequence, the worst case scenario is very unlikely.

7.6 IMPLEMENTATION

A basic demonstrator denoted ADVOKAT-Lottery has been implemented to carry out a classical lottery (CL). It shares the same JavaScript code base of ADVOKAT that has been used in Section 5.5 for the simulation and for ADVOKAT-Vote in Section 6.3. The authority has been omitted in favour of free participation. The setup of a new lottery, the invitation using a link, and the final result is depicted in Figure 7.1. Confirmation requests are not covered yet, so that in the case of protocol deviations distinct results may occur. At this stage, tests have been run with few players. The code of the implementation is available at `https://gitlab.inria.fr/riemann/advokat-lottery` and licensed under GPL v3.0.

7.7 SUMMARY

A novel online lottery protocol is described that relies on a distributed random process carried out by all players in a peer-to-peer manner. Players are assumed to participate throughout the random process. Unlike Chow's protocol (Chow et al., 2005), it allows for both classic lottery and lotto. It provides correctness and verification of the random process based on the assumption of a well-distributed minority of dishonest players. In the most likely scenario, the correctness of the random process is based on an information theoretical secure sharing scheme instead of assumptions on the communication or computational capacities of the authority or the adversary. Further, cryptography has been reduced to asymmetric encryption and signatures. As in many distributed protocols (Nakamoto, 2008; Gambs et al., 2011), the provided security is probabilistic, which may be acceptable for a lottery. The security of the lottery depends on the underlying aggregation algorithm, that needs further examination to quantify the impact of adversaries.



(a) Setup Phase (b) Preparation Phase (c) Evaluation Phase

Figure 7.1: Screenshot of the application ADVOKAT-Lottery. It has been implemented using HTML and JavaScript to run in the browser. In (a), a player creates a new lottery. During the preparation phase (b), other players can be invited using the link, that contains the preparation phase deadline and an initial contact to join the network. After the aggregation phase, the lottery winner is computed using the closest distance to the aggregate ID of the root aggregate (c).

8 CONCLUSION

In this thesis, protocols for secure aggregation of confidential data and for voting in particular have been studied for their trust assumptions. Therefore, protocols were covered starting from the show of hands in Ancient Greece, to paper-based and mechanical solutions in the 19th and 20th century, and eventually to electronic as well as online voting.

Over time, societies have been subject to change and and thus requirements on votings or elections to be considered trustworthy. While in the 19th century an oath on the Bible was considered an adequate mean to ensure trustworthiness, most democracies require nowadays universally verifiable elections that do not rely on institutional or technological trust. However, all commercially available online voting applications fail to meet these requirements, which is why in most cases online voting is not offered.

The distributed protocols BitBallot and ADVOKAT provide a middleware for the secure aggregation of confidential data. They ensure a high degree of confidentiality and rely for this neither on trusted third parties nor solely on cryptography. Instead, the link between data and their sources is obfuscated due to randomly predetermined routing and hierarchical in-network aggregation. The complexity inherent to advanced cryptography is traded for a complexity in communication and assumptions on the trustworthiness of the network as a whole. With no trusted authorities and less reliance on technological trust, both distributed protocols are promising for online voting. This has been claimed in the thesis statement in Section 1.2 that we confirm.

Obviously, all privacy-preserving online aggregations rely on communication technology and require therefore either a basis of technological trust or a basis of technological expertise. However, one can suppose that in future the trust in technology or the technological proficiency evolves, so that distributed online voting is considered trustworthy and eventually employed. For an application in practice, a number of assumptions on the IT infrastructure must be met. In closing, light is shed on the legal and social perspective on consequences and significance of distributed protocols for trustworthiness.

TECHNOLOGICAL ASSUMPTIONS The protocol must be implemented in code to be executed on a platform, e. g. a computer, smartphone, or a cloud service. If the implementation or the platform is compromised, the protocol security is at stake. The verifiable security of such platforms is an open research question. Approaches such as the standardised *Trusted Platform Module* (Fink, Sherman and Carback, 2009) rely

often on cryptography, thus, increase the need for technological trust. Yet, distributed protocols allow for great diversity in the choice of compatible implementations and platforms. The impact of flawed code or platforms is then limited to their respective installation base.

Unlike classical online voting protocols such as Helios, distributed protocols assume that peers are responsive throughout the aggregation. However, in the model of the Internet of Things, permanent connectivity is a common requirement.

LEGAL IMPLICATIONS Many governments regulate protocols used to elect representative bodies. The precision level of regulations spans from general notions of secrecy and verifiability in constitutions (*German Basic Law* 2014) to detailed guidelines on the encryption in distinguished provisions (Collard and Fabre, 2014). Generally, regulations fall short to describe the acceptable risk of defects. While risks due to *random errors*, e. g. inaccurate ballot counting by hand (Goggin et al., 2012) or malicious or accidental destruction of paper ballots prior to counting, are accepted, the legal evaluation of *systematic errors* is yet unclear, but relevant for ADVOKAT. The leak of initial aggregates in ADVOKAT can be interpreted as a quantifiable systematic error of the secrecy.

Suppose that the risk due to systematic errors of ADVOKAT is found to be smaller than the risk due to random errors of alternative protocols with no systematic error. This situation may occur whenever the alternative protocol entails a massive global damage in the worst case scenario, e. g. the leak of all secret ballots in a voting due to a malicious server attack. Shall individual rights on perfect ballot secrecy allowed to be traded for an improved global confidentiality? A similar question with focus on anonymisation has been addressed by Hartzog and Rubinstein (2017) who argue for an assessment based on risk.

'Those who would give up essential Liberty, to purchase a little temporary Safety, deserve neither Liberty nor Safety.'
—Benjamin Franklin

SOCIAL IMPLICATIONS Distributed protocols, and as such BitBallot and ADVOKAT, offer interesting properties for voting. Due to the resources contributed by all voters, everyone may call for such an *affordable* large-scale voting once the electorate is registered. New forms of civic participation with official or unofficial referendums may emerge.

Furthermore, distributed votings are *uninterruptible* after registration and cannot be inhibited, e. g. due confiscation of ballots and ballot boxes, or barricading of polling stations as happened in Spain (Dowsett, 2017; Orihuela, 2017), or DDoS attacks in the case of classical online voting.

Centralisation for the benefit of efficiency has been crucial to cope with cooperation of increasing size, such as in industrial monoculture farming, urbanisation, social networks, etc. However, the advantages of diversification and decentralisation are recognised in an increasing number of domains. The question to which degree decentralisation is beneficial for governance or democracy is yet to be addressed.

A

ANALYSIS OF THE BITBALLOT PULL-PRINCIPLE

Many distributed aggregation protocols such as DPol and SPP (cf. Section 3.1.5.1, respectively Section 3.1.6.1) rely on peers that forward their own intermediate aggregates. BitBallot, introduced in Chapter 4, uses instead the pull principle. Every peer requests pro-actively information from other peers. Requests may be responded with own aggregates or foreign aggregates from previous communication with other peers. The assessment in Section 4.5 is based on the following analysis.

A.1 SCALABILITY

In order to evaluate the scalability of BitBallot, a statistical analysis is presented to determine the number of RPCs t of a peer at node N to acquire all distinct aggregates of the $k-1$ sibling nodes of N for a tree with arity k. This effort is required for each intermediate aggregate, hence $h-1$ times for a tree of height h.

The pull principle introduced in Section 4.2.1 allows peers to respond RPCs with aggregates of other sibling nodes, so that multiple requests to potentially distinct peers may be responded with the identical aggregate. In the best case, each RPC to a distinct peer is responded with a yet unknown aggregate, because those requested peers have not yet learned any other aggregate than their own and belong all to dinstinct sibling subtrees. Only $k-1$ RPCs would be required. This situation may occur if one peer carries out RPCs much faster than others. In the worst case, all other peers belonging to sibling nodes know $k-1$ aggregates and respond with one arbitrary. This situation may occur if one peer starts the aggregation much later or executes the protocol much slower.

The worst case is not representative, but the situation is approached naturally as the aggregation to compute one intermediate aggregate advances. For the following analysis, this worst case is assumed. In this model, a peer P_i belonging to node N requests other peers and may receive an aggregate only of a peer P_j belonging to a sibling node of N. The responses exhibit a uniform distribution of the $k-1$ aggregates. To ease the computation, it is assumed that P_i or potential other peers belonging to N do not provide the k-th aggregate. The following notation is employed:

t	number of RPCs send by P_i
r	number of non-empty responses after t RPCs, $r \leq t$
n	number of peers

n'	number of peers without P_i, $n' = n - 1$
k	arity of the tree
k'	number of sibling nodes, $k' = k - 1$
l	number of acquired (foreign) distinct aggregates of P_i

k^{h-d} peers belong to the subtree with root node N with depth d. N has k' sibling nodes with subtrees of the same size, hence with $k'k^{h-d}$ peers that may provide an aggregate for the sibling nodes of N. The probability to choose any of that peer denoted P_j out of all n' other peers is $p_s = k'k^{h-d}/n'$ and for leaf nodes $p_s = k'/n'$.

The probability $p_r(t)$ that P_i gets exactly r responses for t requests is given with the Bernoulli distribution $f(r, t, p_s)$. p_s is the probability of the event, that a request is responded.

$$p_r(t) = f(r, t, p_s)$$

Further, we have to consider the probability $p_{r,l}$ to have l distinct aggregates in the set of r responses. The k' potential responses are assumed to be uniformly distributed. Based on Table A.1, we can deduce the following formula[1]:

$$p_{r,l} = p_{r-1,l}\frac{l}{k'} + p_{r-1,l-1}\left(1 - \frac{l-1}{k'}\right) \tag{A.1}$$

A probability matrix is plotted in Figure 4.4a.

A.2 CONFIDENTIALITY

In the context introduced above, consider now that any P_j requests the peer P_i after P_i has already carried out t requests. $p(t)$ is defined as the

1 A closed form may be developed on the basis of binomial coefficients that provide for a similar relation: $\binom{r}{l} = \binom{r-1}{l} + \binom{r-1}{l-1}$

r	l	$p_{r,l}$	DESCRIPTION
1	1	1	single elements are always distinct
2	1	k'^{-1}	find two times the same a_i
2	2	$1 - k'^{-1}$	find a different 2nd element
r	$l > r$	0	
r	0	0	
0	1	0	

Table A.1: Probability $p_{r,l}$ to have l distinct acquired aggregates in the set of responses with cardinality of r.

probability that P_i responds to a request with its aggregate. Assuming that every requests requires a fixed amount of time, $p(t)$ can be interpreted as the probability development in time.

$$p(t) = f(0, t, p_s) + \sum_{r=1}^{t} f(r, t, p_s) \sum_{l=1}^{\min\{k', r\}} \frac{p_{r,l}}{1+l} \tag{A.2}$$

The first term accounts for the case that no answer ($r = 0$) has been received, so there is no othen choice as to respond with its aggregate. The second term describes the probability to respond with its aggregate despite having received l distinct aggregates of sibling nodes in the set of r non-empty responses after t requests. An example of $p(t)$ for some configuration is provided in Figure 4.4b.

BIBLIOGRAPHY

Abe, Masayuki and Tatsuaki Okamoto (2000). 'Provably Secure Partially Blind Signatures'. In: *Advances in Cryptology—CRYPTO 2000*. Ed. by Mihir Bellare, pp. 271–286. DOI: 10.1007/3-540-44598-6_17.

Adida, Ben (2006). 'Advances in Cryptographic Voting Systems'. PhD thesis. Massachusetts institute of technology. URL: http://assets.adida.net/research/phd-thesis.pdf.

— (2008). 'Helios: Web-based Open-Audit Voting.' In: *USENIX Security Symposium* 17, pp. 335–348. URL: https://www.usenix.org/legacy/event/sec08/tech/full_papers/adida/adida.pdf.

Adida, Ben, O De Marneffe, Olivier Pereira and JJ Quisquater (2009). 'Electing a university president using open-audit voting: Analysis of real-world use of Helios'. In: *EVT/WOTE* 9. URL: http://dl.acm.org/citation.cfm?id=1496711.1496734.

Aditya, Riza (2005). 'Secure electronic voting with flexible ballot structure'. PhD thesis. Queensland University of Technology. URL: http://eprints.qut.edu.au/16156/.

Aleksander Essex (2013). 'Cryptographic End-to-end Verifiability for Real-world Elections'. Doctoral Thesis. University of Waterloo, p. 272. URL: www.chaum.com/projects/scantegrity/thesis/Essex_Aleks.pdf.

Ali, Syed Taha and Judy Murray (2016). 'An Overview of End-to-End Verifiable Voting Systems'. In: *Real-World Electronic Voting: Design, Analysis and Deployment*. Ed. by Feng Hao and Peter Y. A. Ryan. CRC Press. arXiv: 1605.08554.

Androutsellis-Theotokis, Stephanos and Diomidis Spinellis (2004). 'A Survey of Peer-to-peer Content Distribution Technologies'. In: *ACM Comput. Surv.* 36.4, pp. 335–371. ISSN: 0360-0300. DOI: 10.1145/1041680.1041681.

Arora, Siddhartha (2008). 'National e-ID card schemes: A European overview'. In: *Information Security Technical Report* 13.2, pp. 46–53. DOI: 10.1016/j.istr.2008.08.002.

Baumgart, Ingmar and Sebastian Mies (2007). 'S/Kademlia: A practicable approach towards secure key-based routing'. In: *ICPADS '07: Proceedings of the 13th International Conference on Parallel and Distributed Systems*. IEEE Computer Society. Washington, DC, USA: IEEE Computer Society, pp. 1–8. DOI: 10.1109/ICPADS.2007.4447808.

Benet, Juan (2014). 'IPFS-Content Addressed, Versioned, P2P File System'. In: *arXiv preprint arXiv:1407.3561* Draft 3. arXiv: arXiv:1407.3561v1. URL: http://ipfs.io.

Bernhard, Matthew, Josh Benaloh, J. Alex Halderman, Ronald L. Rivest, Peter Y. A. Ryan, Philip B. Stark, Vanessa Teague, Poorvi L. Vora and Dan S. Wallach (2017). 'Public Evidence from Secret Ballots'. In: arXiv: 1707.08619.

Boegehold, Alan L (1963). 'Toward a Study of Athenian Voting Procedure'. In: *Hesperia* 32.4, p. 366. DOI: 10.2307/147360.

Boucher, Philip (2016). *What if blockchain technology revolutionised voting?* Tech. rep. European Parliamentary Research Service. URL: http://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_ATA(2016)581918.

Brightwell, Ian, Jordi Cucurull, David Galindo and Sandra Guasch (2015). *An overview of the iVote 2015 voting system*. Tech. rep. New South Wales Electoral Commission, pp. 1–25. URL: https://www.elections.nsw.gov.au/__data/assets/pdf_file/0019/204058/An_overview_of_the_iVote_2015_voting_system_v4.pdf.

Cai, Xing Shi and Luc Devroye (2013). 'A probabilistic analysis of Kademlia networks'. In: *Proc. of International Symposium on Algorithms and Computation*. Vol. 8283 LNCS. ISAAC '13. Springer, pp. 711–721. DOI: 10.1007/978-3-642-45030-3_66. arXiv: 1402.1191.

Casanova, Henri, Arnaud Giersch, Arnaud Legrand, Martin Quinson and Frédéric Suter (2014). 'Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms'. In: *Journal of Parallel and Distributed Computing* 74.10, pp. 2899–2917. oai:hal.inria.fr:hal-01017319. URL: http://simgrid.gforge.inria.fr.

Chang-Fong, N. and Aleksander Essex (2016). 'The cloudier side of cryptographic end-to-end verifiable voting: A security analysis of helios'. In: *32nd Annual Computer Security Applications Conference*. ACM Press, pp. 324–335. DOI: 10.1145/2991079.2991106.

Chaum, David (1981). 'Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms'. In: *Communications of the ACM* 24.2, pp. 84–90. DOI: 10.1145/358549.358563.

— (1983). 'Blind Signatures for Untraceable Payments'. In: *Advances in Cryptology: Proceedings of CRYPTO '82*. Ed. by David Chaum, Ronald L. Rivest and Alan T. Sherman. Boston, MA: Springer US, pp. 199–203. DOI: 10.1007/978-1-4757-0602-4_18.

Chevallier-Mames, Benoît, Pierre-Alain Fouque, David Pointcheval, Julien Stern and Jacques Traoré (2010). 'On Some Incompatible Properties of Voting Schemes'. In: *Towards Trustworthy Elections: New Directions in Electronic Voting*. Berlin, Heidelberg: Springer, pp. 191–199. DOI: 10.1007/978-3-642-12980-3_11.

Chor, Benny, Shafi Goldwasser, Silvio Micali and Baruch Awerbuch (1985). 'Verifiable secret sharing and achieving simultaneity in the presence of faults'. In: *26th Annual Symposium on Foundations of Computer Science*. FOCS'85. IEEE, pp. 383–395. DOI: 10.1109/SFCS.1985.64.

Chow, Sherman S M, Lucas C K Hui, S M Yiu and K P Chow (2005). 'An e-Lottery Scheme Using Verifiable Random Function'. In: *Proc. ICCSA 2005*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 651–660. DOI: 10.1007/11424857_72.

Chowdhury, Areeq (2014). *Viral Voting: Future-proofing UK elections with an #onlinevoting option*. Tech. rep. WebRoots Democracy, p. 95. URL: https://webrootsdemocracy.org/viral-voting/.

Cisco Systems (2008). *Cisco Visual Networking Index: Approaching the Zettabyte Era*. Tech. rep.

Coggins, Jay S. and C. Federico Perali (1998). '64% Majority rule in Ducal Venice: Voting for the Doge'. In: *Public Choice* 97.4, pp. 709–723. ISSN: 00485829. DOI: 10.1023/A:1004947715017.

Cohen, Bram (2003). *Incentives Build Robustness in BitTorrent*. URL: http://bittorrent.org/bittorrentecon.pdf (visited on 22/05/2017).

— (2008). *The BitTorrent Protocol Specification*. URL: http://bittorrent.org/beps/bep_0003.html (visited on 03/05/2017).

Collard, Susan and Elodie Fabre (2014). 'Electronic Voting in the French Legislative Elections of 2012'. In: *Design, Development, and Use of Secure Electronic Voting Systems*. IGI Global, pp. 176–198. DOI: 10.4018/978-1-4666-5820-2.ch009.

Colmar Brunton Social Research Agency (2012). *Voter and Non-Voter Satisfaction Survey 2011*. report. Takapuna, Auckland: Australian Electoral Commisson, p. 130. URL: http://www.elections.org.nz/events/past-events-0/2011-general-election/reports-and-surveys-2011-general-election/voter-and-non.

Comparative Data (2014). Administration and Cost of Elections Project. URL: http://aceproject.org/epic-en (visited on 21/10/2016).

Constitution of France (1958). Constitutional Council. URL: http://www.conseil-constitutionnel.fr (visited on 24/04/2017).

Cramer, Ronald, Rosario Gennaro and Berry Schoenmakers (1997). 'A Secure and Optimally Efficient Multi-Authority Election Scheme'. In: *Proc. of International Conference on the Theory and Application of Cryptographic Techniques*. Vol. 1233. EUROCRYPT '97. Springer, pp. 103–118. DOI: 10.1007/3-540-69053-0_9.

Dill, David L et al. (2012). *Computer Technologists' Statement on Internet Voting*. Opinion. Verified Voting Foundation. URL: https://www.verifiedvoting.org/projects/internet-voting-statement/.

Dingledine, Roger, Nick Mathewson and Paul Syverson (2004). 'Tor: The second-generation onion router'. In: *SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium* 13, p. 21. DOI: 10.1.1.4.6896.

Document of the Copenhagen Meeting (1990). Copenhagen: Organization for Security and Co-operation in Europe. URL: http://www.osce.org/odihr/elections/14304 (visited on 25/04/2017).

Dowsett, Sonya (2017). *Police in Catalonia hunt for hidden ballot boxes in bid to foil referendum*. URL: https://www.reuters.com/article/idUSKCN1BQ2MZ (visited on 28/09/2017).

Driza Maurer, Ardita and Jordi Barrat, eds. (2015). *E-voting case law : a comparative analysis*. Taylor & Francis Ltd., p. 324. ISBN: 9781472446756.

Election Process Advisory Commission (2007). *Voting with confidence*. The Hague: Netherlands Election Process Advisory Commission. URL: http://wijvertrouwenstemcomputersniet.nl/images/0/0c/Votingwithconfidence.pdf.

European Convention on Human Rights (1950). Rome: Council of Europe. URL: http://www.echr.coe.int/Documents/Convention_ENG.pdf (visited on 25/04/2017).

Facebook (2017). *Company Info*. URL: https://newsroom.fb.com/company-info/ (visited on 18/09/2017).

Federal Electoral Regulations (2017). German Federal Returning Officer. URL: https://www.bundeswahlleiter.de/en/bundestagswahlen/2017/rechtsgrundlagen.html (visited on 25/04/2017).

Fink, R.A., A.T. Sherman and R. Carback (2009). 'TPM Meets DRE: Reducing the Trust Base for Electronic Voting Using Trusted Platform Modules'. In: *IEEE Transactions on Information Forensics and Security* 4.4, pp. 628–637. ISSN: 1556-6013. DOI: 10.1109/TIFS.2009.2034900.

Fouque, Pierre-Alain, Guillaume Poupard and Jacques Stern (2001). 'Sharing Decryption in the Context of Voting or Lotteries'. In: *Proceedings of the 4th International Conference on Financial Cryptography*. Berlin, Heidelberg: Springer, pp. 90–104. DOI: 10.1007/3-540-45472-1_7.

Frénot, Stéphane, Stéphane Grumbach and Damien Reimert (2013). 'A P2P Protocol for Privacy Preserving Cooperative Decision Making'. Unpublished private communication.

— (2014). 'E-Voting, the Case for Decentralised Systems'. In: *Proc. of Cooperative Technologies in Democratic Processes - Beyond e-Voting*. COOP 2014. Nice. oai:hal.inria.fr:hal-01097479.

Fujioka, Atsushi, Tatsuaki Okamoto and Kazuo Ohta (1993). 'A Practical Secret Voting Scheme for Large Scale Elections'. In: *Proc. of Advances in Cryptology*. Vol. 718. AUSCRYPT'92. Berlin, Heidelberg: Springer, pp. 244–251. DOI: 10.1007/3-540-57220-1.

Gambs, Sebastien, Rachid Guerraoui, Hamza Harkous, Florian Huc and Anne-Marie Kermarrec (2011). 'Scalable and Secure Aggregation in Distributed Networks'. In: *31st Symposium on Reliable Distributed Systems*. Irvine, CA: IEEE, pp. 181–190. DOI: 10.1109/SRDS.2012.63.

German Basic Law (2014). Berlin: Federal Republic of Germany. URL: http://www.gesetze-im-internet.de/englisch_gg/ (visited on 25/04/2017).

Girard, Charles (2010). 'Acclamation Voting in Sparta: An Early Use of Approval Voting'. In: *Studies in Choice and Welfare*. Ed. by Jean-François Laslier and M. Remzi Sanver. Springer Nature. Chap. 2, pp. 15–17. DOI: 10.1007/978-3-642-02839-7_2.

Goggin, Stephen N, Michael D Byrne and Juan E Gilbert (2012). 'Post-Election Auditing: Effects of Procedure and Ballot Type on Manual Counting Accuracy, Efficiency, and Auditor Satisfaction and Confidence'. In: *Election Law Journal: Rules, Politics, and Policy* 11.1, pp. 36–51. DOI: 10.1089/elj.2010.0098.

Goldschlag, David M and Stuart G Stubblebine (1998). 'Publicly verifiable lotteries: Applications of delaying functions'. In: *Proc. of 2nd International Conference on Financial Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 214–226. DOI: 10.1007/BFb0055485.

Guerraoui, Rachid, Kévin Huguenin, Anne Marie Kermarrec, Maxime Monod and Ýmir Vigfsson (2012). 'Decentralized polling with respectable participants'. In: *Journal of Parallel and Distributed Computing* 72.1, pp. 13–26. ISSN: 07437315. DOI: 10.1016/j.jpdc.2011.09.003. URL: https://hal.inria.fr/inria-00629455.

Haenni, Rolf and Oliver Spycher (2011). 'Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys'. In: *Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*. EVT/WOTE'11. Berkeley, CA, USA: USENIX Association, p. 8. URL: http://dl.acm.org/citation.cfm?id=2028012.2028020.

Halderman, J. Alex and Vanessa Teague (2015). 'The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election'. In: *Proc. of the 5th International Conference on E-Voting and Identity*. VoteID'15. Bern: Springer, pp. 35–53. DOI: 10.1007/978-3-319-22270-7_3. arXiv: 1504.05646.

Hansen, Mogens Herman (1977). 'How Did the Athenian Ecclesia Vote?' In: *Greek, Roman and Byzantine Studies* 18.2, p. 123. URL: http://grbs.library.duke.edu/issue/view/1451.

Hartzog, Woodrow and Ira Rubinstein (2017). 'The Anonymization Debate Should Be About Risk, Not Perfection'. In: *Commun. ACM* 60.5, pp. 22–24. DOI: 10.1145/3068787.

Hoang, Bao-Thien and Abdessamad Imine (2015). 'Efficient and Decentralized Polling Protocol for General Social Networks'. In: *Stabilization, Safety, and Security of Distributed Systems: 17th International Symposium*. Ed. by Andrzej Pelc and Alexander A. Schwarzmann. Springer International Publishing, pp. 171–186. DOI: 10.1007/978-3-319-21741-3_12.

Horwitz, Sari (2016). *More than 30 states offer online voting, but experts warn it isn't secure*. The Washington Post. URL: http://wapo.st/1XgqtqS (visited on 19/04/2017).

Ibrahim, Maged Hamada (2017). 'SecureCoin: A Robust Secure and Efficient Protocol for Anonymous Bitcoin Ecosystem'. In: *Interna-*

tional Journal of Network Security 19.2, pp. 295–312. DOI: 10.6633/IJNS.201703.19(2).14.

Isidore, Chris (2015). *Americans spend more on the lottery than on ...* URL: http://money.cnn.com/2015/02/11/news/companies/lottery-spending/ (visited on 16/02/2017).

Jones, Douglas W. (2001). *Voting and Elections*. URL: http://homepage.divms.uiowa.edu/~jones/voting/ (visited on 04/04/2017).

Jonker, Hugo, Sjouke Mauw and Jun Pang (2013). 'Privacy and verifiability in voting systems: Methods, developments and trends'. In: *Computer Science Review* 10, pp. 1–30. DOI: 10.1016/j.cosrev.2013.08.002.

Khan, Rafiullah, Sarmad Ullah Khan, Rifaqat Zaheer and Shahid Khan (2012). 'Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges'. In: *Proc. of the 10th International Conference on Frontiers of Information Technology*. FIT'12. IEEE. DOI: 10.1109/fit.2012.53.

Kitsing, Meelis (2011). 'Online Participation in Estonia: Active Voting, Low Engagement'. In: *Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance*. ICEGOV '11. New York: ACM, pp. 20–26. DOI: 10.1145/2072069.2072073.

Kuacharoen, Pramote (2012). 'Design and Implementation of a Secure Online Lottery System'. In: *Proc. IAIT 2012*. Springer Berlin Heidelberg, pp. 94–105. DOI: 10.1007/978-3-642-35076-4_9.

Lambrinoudakis, Costas, Dimitris Gritzalis, Vassilis Tsoumas, Maria Karyda and Spyros Ikonomopoulos (2003). 'Secure electronic voting: The current landscape'. In: *Secure Electronic voting*. Ed. by Dimitris A. Gritzalis. Springer, pp. 101–122. DOI: 10.1007/978-1-4615-0239-5_7.

Lamport, Leslie (1998). 'The part-time parliament'. In: *ACM Transactions on Computer Systems* 16.2, pp. 133–169. DOI: 10.1145/279227.279229.

Lamport, Leslie, Robert Shostak and Marshall Pease (1982). 'The Byzantine Generals Problem'. In: *ACM Transactions on Programming Languages and Systems* 4.3, pp. 382–401. DOI: 10.1145/357172.357176.

Lenstra, Arjen K and Benjamin Wesolowski (2015). 'A random zoo: sloth, unicorn, and trx'. In: *NIST Workshop on Elliptic Curve Cryptography Standards* 3.

Liu, Yining, Lei Hu and Heguo Liu (2007). 'Using an Efficient Hash Chain and Delaying Function to Improve an e-Lottery Scheme'. In: *Int. J. Comput. Math.* 84.7, pp. 967–970. DOI: 10.1080/00207160701294426.

Locher, Philipp and Rolf Haenni (2016). 'Receipt-free remote electronic elections with everlasting privacy'. In: *Annals of Telecommunications* 71.7-8, pp. 323–336. DOI: 10.1007/s12243-016-0519-6.

Loewenstern, Andrew (2008). *DHT Protocol*. URL: http://www.bittorrent.org/beps/bep_0005.html (visited on 04/05/2017).

MacWilliams, Matthew C (2015). 'Forecasting Congressional Elections Using Facebook Data'. In: *PS: Political Science & Politics* 48.04, pp. 579–583. DOI: 10.1017/s1049096515000797.

Markowitch, Olivier and Jérôme Dossogne (2010). 'E-voting : Individual verifiability of public boards made more achievable'. In: *31th Symposium on Information Theory in the Benelux*. Rotterdam: Werkgemeenschap voor Informatie en Communicatietheorie, pp. 5–10.

Mauw, S., J. Verschuren and Erik P. de Vink (2007). 'Data Anonymity in the FOO Voting Scheme'. In: *Electronic Notes in Theoretical Computer Science* 168, pp. 5–28. DOI: 10.1016/j.entcs.2006.11.001.

Maymounkov, Petar and D Mazieres (2002). *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*. Ed. by Peter Druschel, Frans Kaashoek and Antony Rowstron. Vol. 2429. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 53–65. DOI: 10.1007/3-540-45748-8. arXiv: 1011.1669v3.

McCorry, Patrick, Siamak F. Shahandashti and Feng Hao (2017). *A Smart Contract for Boardroom Voting with Maximum Voter Privacy*. Cryptology ePrint Archive, Report 2017/110. URL: https://eprint.iacr.org/2017/110.

Merkle, Ralph C. (1988). 'A Digital Signature Based on a Conventional Encryption Function'. In: *Proc. of Advances in Cryptology*. Ed. by Carl Pomerance. CRYPTO'87. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 369–378. DOI: 10.1007/3-540-48184-2_32.

Miers, Ian, Christina Garman, Matthew Green and Aviel D. Rubin (2013). 'Zerocoin: Anonymous distributed e-cash from bitcoin'. In: *Proc. IEEE Symposium on Security and Privacy*, pp. 397–411. ISBN: 9780769549774. DOI: 10.1109/SP.2013.34.

Mowbray, Miranda and Dieter Gollmann (2007). 'Electing the Doge of Venice: Analysis of a 13th century protocol'. In: *Computer Security Foundations* 28.July, pp. 6–8. DOI: 10.1109/CSF.2007.21.

Nakajima, Amane (1993). 'Decentralized Voting Protocols'. In: *Proc. of International Symposium on Autonomous Decentralized Systems*. ISAD'93. Kawasaki: IEEE, pp. 247–254. DOI: 10.1109/ISADS.1993.262697.

Nakamoto, Satoshi (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf.

Orihuela, Rodrigo (2017). *The Ground War Over Catalonia Is Being Fought in Cyberspace*. URL: https://www.bloomberg.com/news/articles/2017-09-30/the-ground-war-over-catalonia-is-being-fought-in-cyberspace (visited on 30/09/2017).

Pedersen, Torben Pryds (1991). 'A threshold cryptosystem without a trusted party'. In: *Advances in Cry*, pp. 522–526. DOI: 10.1016/j.amc.2004.01.018.

Perez-Marco, Ricardo (2016). 'Bitcoin and Decentralized Trust Protocols'. In: *Newsletter of the EMS*, pp. 1–8. arXiv: 1601.05254.

Pierrot, Cecile and Benjamin Wesolowski (2016). 'Malleability of the blockchain's entropy'. In: *ArcticCrypt 2016*, pp. 1–20. URL: http://eprint.iacr.org/2016/370.

Ptacek, Thomas (2011). *Javascript Cryptography Considered Harmful*. NCC Group. (Visited on 26/09/2017).

Rabin, T and M Ben-Or (1989). 'Verifiable Secret Sharing and Multiparty Protocols with Honest Majority'. In: *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*. STOC '89. New York, NY, USA: ACM, pp. 73–85. DOI: 10.1145/73007.73014.

Reimert, Damien, Stéphane Frénot, Stéphane Grumbach and Simon Meyffret (2016). 'Machine de Vote électronique et Infrastructure comportant une telle Machine'. Patent FR 3037702 (France). URL: http://bases-brevets.inpi.fr/en/document-en/FR3037702.

Reynolds, Andrew, Ben Reilly and Andrew Ellis (2005). *Electoral System Design: The New International IDEA Handbook*. Stockholm: International IDEA, p. 223. ISBN: 91-85391-18-2. URL: http://www.idea.int/publications/esd/upload/ESD_Handb_low.pdf.

Riemann, Robert and Stéphane Grumbach (2017a). 'Distributed Protocols at the Rescue for Trustworthy Online Voting'. In: *Proc. of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*. Porto, pp. 499–505. ISBN: 978-989-758-209-7. DOI: 10.5220/0006228504990505. URL: https://hal.inria.fr/hal-01501424.

— (2017b). 'Secure and trustable distributed aggregation based on Kademlia'. In: *IFIP Advances in Information and Communication Technology*. Ed. by F. Martinelli and S. De Capitani di Vimercati. Vol. 502. Rome: Springer. Chap. 12, pp. 171–185. ISBN: 978-3-319-58468-3. DOI: 10.1007/978-3-319-58469-0_12. URL: https://hal.inria.fr/hal-01529326.

— (2017c). 'Distributed Random Process for a large-scale Peer-to-Peer Lottery'. In: *Proc. of 17th IFIP Distributed Applications and Interoperable Systems*. DAIS'17. Neuchâtel: Springer, pp. 34–48. ISBN: 978-3-319-59664-8. DOI: 10.1007/978-3-319-59665-5_3. URL: https://hal.inria.fr/hal-01583824.

Riera, Andreu (1998). 'An Introduction to Electronic Voting Schemes'. In: pp. 1–22. URL: http://pirdi.uab.es/document/pirdi9.ps.

Rivest, Ronald L. (2008). 'On the notion of 'software independence' in voting systems'. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 366.1881, pp. 3759–3767. DOI: 10.1098/rsta.2008.0149.

Rivest, Ronald L. and Warren D. Smith (2007). 'Three Voting Protocols : ThreeBallot, VAV, and Twin ThreeBallot'. In: *Proc. of the USENIX Workshop on Accurate Electronic Voting Technology*, pp. 16–16.

Rodgers, Grant (2015). *Guilty verdict in Hot Lotto scam, but game safe, official says*. Des Moines, USA. URL: http://dmreg.co/1JbGgRN (visited on 23/01/2017).

Saltman, Roy G. (2006). *The History and Politics of Voting Technology*. Springer. DOI: 10.1057/9781403977212.

Schneier, Bruce (2001). *Crypto-Gram (Februray)*. URL: https://www.schneier.com/crypto-gram/archives/2001/0215.html#10 (visited on 06/04/2017).

— (2012). *Liars and outliers : enabling the trust that society needs to thrive*. Wiley, p. 384. ISBN: 1118143302.

— (2016). *By November, Russian Hackers Could Target Voting Machines*. The Washington Post. URL: http://wapo.st/2a5c9NF (visited on 19/04/2017).

Shackelford, Scott, Bruce Schneier, Michael Sulmeyer, Anne E. Boustead, Ben Buchanan, Amanda Craig, Trey Herr and Jessica Zhanna Malekos Smith (2016). 'Making Democracy Harder to Hack: Should Elections Be Classified as 'Critical Infrastructure?'' In: *University of Michigan Journal of Law Reform*. URL: https://ssrn.com/abstract=2852461.

Shamir, Adi (1979). 'How to Share a Secret'. In: *Communications of the ACM* 22.11, pp. 612–613. DOI: 10.1145/359168.359176.

Simons, Barbara (2004). 'Electronic Voting Systems: the Good, the Bad, and the Stupid'. In: *ACM Queue* November 2003, pp. 20–26. DOI: 10.1145/1035594.1035606.

Simons, Barbara and Douglas W. Jones (2012). 'Internet Voting in the U.S.' In: *Commun. ACM* 55.10, pp. 68–77. DOI: 10.1145/2347736.2347754.

Springall, Drew, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine and J Alex Halderman (2014). 'Security Analysis of the Estonian Internet Voting System'. In: *Proceedings of the 21st ACM Conference on Computer and Communications Security*. ACM, pp. 703–715. DOI: 10.1145/2660267.2660315.

Swanson, Colleen M and Douglas R Stinson (2011). 'Unconditionally Secure Signature Schemes Revisited'. In: *Information Theoretic Security: 5th International Conference, ICITS 2011. Proceedings*. Ed. by Serge Fehr. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 100–116. DOI: 10.1007/978-3-642-20728-0_10.

Thielman, Sam and Chris Johnston (2016). *Major cyber attack disrupts internet service across Europe and US*. URL: https://www.theguardian.com/technology/2016/oct/21/ddos-attack-dyn-internet-denial-service (visited on 01/10/2017).

Thomas, Leigh and John Stonestreet (2017). *France drops electronic voting for citizens abroad over cybersecurity fears*. URL: https://www.reuters.com/article/idUSKBN16D233 (visited on 25/09/2017).

Traffic & Figures (2017). Worldwide Interbank Financial Telecommunication. URL: https://www.swift.com (visited on 04/05/2017).

Ülle, Madise and Tarvi Martens (2006). 'E-voting in Estonia 2005. The first practice of country-wide binding Internet voting in the world'. In: *Electronic Voting 2006: 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting.CC*. Octo-

ber. Bonn: Gesellschaft für Informatik, p. 253. URL: https://cs.emis.de/LNI/Proceedings/Proceedings86/GI-Proceedings-86-1.pdf.

Universal Declaration of Human Rights (1948). Paris: Universal Declaration of Human Rights. URL: http://www.un.org/en/universal-declaration-human-rights/.

U.S. Public Policy Council (2010). *Issue Brief: Internet Voting and Uniformed and Overseas Citizens Absentee Voters*. Policy Statement. Association of Computing Machinery (ACM). URL: http://usacm.acm.org/evoting/category.cfm.

Volkamer, Melanie (2010). 'Electronic Voting in Germany'. In: *Data Protection in a Profiled World*. Ed. by Serge Gutwirth, Yves Poullet and Paul De Hert. Dordrecht: Springer Netherlands. Chap. 10, pp. 177–189. DOI: 10.1007/978-90-481-8865-9_10.

Wachs, Matthias (2015). 'A Secure and Resilient Communication Infrastructure for Decentralized Networking Applications'. PhD. München: Technische Universität München, p. 250. ISBN: 3-937201-45-9. URL: http://d-nb.info/1069199680.

Wiesner, Stephen (1983). 'Conjugate Coding'. In: *SIGACT News* 15.1, pp. 78–88. DOI: 10.1145/1008908.1008920.

Wilkinson, Shawn, Tome Boshevski, Josh Brandoff and Vitalik Buterin (2014). 'Storj A Peer-to-Peer Cloud Storage Network Files as Encrypted Shards'. URL: https://storj.io/storj2014.pdf.

Wood, Gavin (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. URL: http://gavwood.com/paper.pdf (visited on 09/05/2017).

Yao, Andrew C. (1982). 'Protocols for secure computations'. In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. DOI: 10.1109/SFCS.1982.38.

Yli-Huumo, Jesse, Deokyoon Ko, Sujin Choi, Sooyong Park and Kari Smolander (2016). 'Where Is Current Research on Blockchain Technology?—A Systematic Review'. In: *PLOS ONE* 11.10, pp. 1–27. DOI: 10.1371/journal.pone.0163477.

Zhao, Zhichao and T-H. Hubert Chan (2015). 'How to Vote Privately Using Bitcoin'. In: *IACR Cryptology ePrint Archive* 2015. URL: https://eprint.iacr.org/2015/1007.

Zhou, Jianying and Chunfu Tan (2001). 'Playing Lottery on the Internet'. In: *Proc. of ICICS 2001*. Ed. by Sihan Qing, Tatsuaki Okamoto and Jianying Zhou. Berlin, Heidelberg: Springer, pp. 189–201. DOI: 10.1007/3-540-45600-7_22.

GLOSSARY

election	formal decision-making process by which a population chooses an individual to hold public office. 5, *see* voting
electorate	set of individuals entitled and registered to vote 5, 112, 132
electronic voting	also *e-voting*; voting that involves electronic data processing for ballot casting tallying. 3, 14, 23, 131
online voting	also *remote electronic voting*; voting that involves the transfer of electronic ballots from one or more polling stations to another location for tallying. 3, 15, 23, 131
polster	voter's software agent to create, cast and verify electronic ballots on behalf of the voter. 32, 62
public bulletin board	broadcast communication channel with memory. All broadcast information is stored and publicly accessible. Participants have an access to write to specific sections of the board. xiii, 32, 85
secure channel	medium to transfer messages resistant to tampering and overhearing from one party to another. 31, 59, 124
tree configuration	actual association of peers to leaf nodes in a sparse tree in ADVOKAT. 87, 92, 97, 105, 126
turnout	ratio of participation, e. g. in a voting. 24, 77, 80, 81, 111
voting	voting is a method for a group such as a meeting or an electorate to make a decision or express an opinion. 5
voting protocol	technical procedure to carry out a voting. 5, 15, *see* voting
voting system	set of rules for translating sets of preferences as expressed in a voting into the actual voting outcome. 5, 15

DECLARATION

Put your declaration here.

Lyon, September 2017

Robert Riemann

COLOPHON

This document was typeset with XeLaTeX using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL*) are used. The "typewriter" text is typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)