Universally Verifiable Poll-Site Voting Schemes Providing Everlasting Privacy

Vom Fachbereich Informatik der

Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades

Doktor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Inform. Denise Demirel

geboren in Frankfurt am Main.

Referenten:	Prof. Dr. Johannes Buchmann
	Prof. Dr. Jeroen van de Graaf
Tag der Einreichung:	17.10.2013
Tag der mündlichen Prüfung:	03.12.2013
Hochschulkennziffer:	D17

Darmstadt 2014

List of Publications

[1] Denise Demirel, Maria Henning, Jeroen van de Graaf, Peter Y. A. Ryan, and Johannes Buchmann. Prêt à Voter providing everlasting privacy. In *VOTE-ID*, pages 156–175, 2013.

[2] Johannes Buchmann, Denise Demirel, and Jeroen van de Graaf. Towards a publicly-verifiable mix-net providing everlasting privacy. In *Financial Cryptography*, pages 197–204, 2013.

[3] Denise Demirel, Jeroen van de Graaf, and Roberto Araùjo. Improving helios with everlasting privacy towards the public. In *Electronic Voting Technology Workshop / Workshop on Trustworthy Elections - EVT/WOTE 2012*, August 2012.

[4] Denise Demirel and Maria Henning. Legal analysis of privacy weaknesses in poll-site evoting systems. *Jusletter IT - Die Zeitschrift für IT und Recht*, September 2012.

[5] Denise Demirel, Hugo Jonker, and Melanie Volkamer. Random block verification: Improving the norwegian electoral mix-net. In Melanie Volkamer Manuel J. Kripp and Rüdiger Grimm, editors, *5th International Conference on Electronic Voting 2012 (EVOTE2012)*, volume 205 of *LNI - Series of the Gesellschaft für Informatik (GI)*, pages 65–78. Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, Gesellschaft für Informatik, July 2012.

[6] Maria Henning, Denise Demirel, and Melanie Volkamer. Öffentlichkeit vs. Verifizierbarkeit - Inwieweit erfüllt mathematische Verifizierbarkeit den Grundsatz der Öffentlichkeit der Wahl. *Internationalen Rechtsinformatik Symposiums, IRIS 2012*, pages 213–220, 2012.

[7] Denise Demirel, Maria Henning, Peter Y. A. Ryan, Steve Schneider, and Melanie Volkamer. Feasibility analysis of prêt à voter for german federal elections. In *3rd*

international conference on e-voting and identity (VoteID 2011), volume 7187 2012, pages 158–173. Springer, September 2011.

[8] Richard Frankland, Denise Demirel, Jurlind Budurushi, and Melanie Volkamer. Side-channels and evoting machine security: Identifying vulnerabilities and defining requirements. In *International Workshop on Requirements Engineering for Electronic Voting Systems (REVOTE'11)*, pages 37–46, August 2011.

[9] Denise Demirel, Richard Frankland, Darko Popovic, and Melanie Volkamer. Voting software to support election preparation, counting, and tallying. In Peter Parycek, Manuel J. Kripp, and Edelmann Noella, editors, *CeDEM11 Proceedings of the International Conference for E-Democracy and Open Government*, pages 287–300. Edition Donau-Universität Krems, May 2011.

Abstract

Computer based voting brings up huge challenges for technology. On the one hand an electronic voting system has to be transparent enough to allow verification of its correct functioning; on the other hand, it must ensure that these verification procedures do not allow an attacker to violate voter privacy.

Both requirements can be addressed by providing cryptographically secured voting receipts. Each voter cast his or her vote in encoded form and receives a copy of the recorded ballot as receipt. The voters can use these receipts to verify that their vote is contained in the input of the tally. Furthermore, the encoded votes are publicly processed, which allows voters and observers to check that the election outcome has been determined correctly. However, to provide a private and free election, no voter should be able to prove to someone else for whom he or she voted. This must not only be prevented during the election, but also afterwards for an indefinite period of time. Especially with respect to *everlasting privacy* this is not ensured by most verifiable voting systems. If the receipt contains, for instance, the voting decision encrypted using some public key cryptography, an attacker can determine the candidates selected as soon as the underlying computational problem has been solved for the key length chosen.

In this work we provide a summary of privacy weaknesses that may arise in verifiable electronic poll-site voting systems, and we identify and solve open issues. More precisely, we concentrate on the following three questions: (1) How can we show correct anonymization of votes in an efficient and privacy preserving manner using a generic approach? (2) How can we introduce everlasting privacy to mixing and homomorphic tallying based voting schemes? (3) How can we reduce the amount of trust voters have to put in authorities regarding privacy?

In electronic voting so-called *reencryption mix-nets* are used to anonymize votes. These mix-nets shuffles votes in a universally verifiable manner, i.e., they publish some audit information allowing voters and observers to verify that the votes came out as they went in. In practice, mostly generic verification procedures are used to show correctness of this process. However, many of them do not provide an adequate

level of privacy. To address (1), we investigate several proposals and introduce a new protocol that combines existing approaches but improves them with respect to privacy and efficiency.

Another drawback of mixing based voting schemes is that all implementations provide *computational privacy* only. We address (2) by presenting a mix-net that uses a homomorphic and unconditionally hiding commitment scheme to encode the votes and audit data, implying *everlasting privacy*. The correctness of the anonymization process is guaranteed with overwhelming probability, even if all authorities collaborate. An implication of our result is that many current voting systems that use mix-nets can be upgraded to everlasting privacy. Subsequently, we show that this protocol can be applied to *Prêt à Voter* and *Split-Ballot* imposing only minor changes to current implementations.

The same approach is used to introduce everlasting privacy to homomorphic tallying based schemes. The votes are encoded with an unconditionally hiding commitment scheme, they are homomorphically tallied in public, and the result is decoded afterwards. To show that our solution can be applied to poll-site voting, we describe how the *Scratch & Vote* voting system can be improved using our tallying protocol. Again only minor changes to the classical scheme are necessary.

To address (3), the approach of non-personalized receipts is analyzed. If the receipts handed out to the voters do not contain a link to their vote cast, they do not have to put their trust in authorities keeping this association secret. We introduce an electronic ballot box that generates non-personalized receipts using a process that is similar to the anonymization procedure carried out by mix-nets. The correctness of the receipt generation is universally verifiable. Furthermore, our approach improves on existing solutions with respect to correctness and privacy.

Finally, we compare all voting systems that are improved in this work, highlight their advantages and disadvantages, and conclude with key issues for future work.

Zusammenfassung

Die Entwickelung elektronisch unterstützter Wahlsysteme stellt eine große Herausforderung für die Forschung dar. Einerseits müssen die Systeme transparent genug sein, um Wählern und Beobachtern die Möglichkeit zu geben die Korrektheit der Ergebnisermittlung zu überprüfen. Andererseits darf das Verifizierungsverfahren Angreifern nicht erlauben das Wahlgeheimnis zu verletzen.

Ein Ansatz für die Umsetzung beider Anforderungen ist die Generierung von kodierten Belegen während der Stimmabgabe. Ein Beleg erlaubt dem Wähler beziehungsweise der Wählerin nach Schließung der Wahlurne zu überprüfen, ob die eigene Stimme richtig von dem System erkannt und hierin gespeichert wurde. Anschließend werden alle kodierten Stimmen öffentlich ausgezählt, wodurch sich Wähler_innen und Beobachter_innen vergewissern können, dass das Wahlergebnis richtig ermittelt wurde. Um eine geheime und freie Wahl zu gewährleisten muss bei der Gestaltung des Belegs insbesondere darauf geachtet werden, dass kein_e Wähler_in in der Lage ist einem Dritten gegenüber zu beweisen wie er oder sie gewählt hat. Dies muss nicht nur während der Wahl verhindert werden, sondern auch im Anschluß an die Wahl. Insbesondere in Bezug auf die dauerhafte Geheimhaltung der abgegebenen Stimme wird das Wahlgeheimnis von vielen Systemen nicht ausreichend gewährleistet. Ein weit verbreiteter Ansatz ist die Stimme mit einem asymmetrischen Verschlüsselungsverfahren zu verschlüsseln. Dies erlaubt jedoch einem Angreifer beziehungsweise einer Angreiferin die getroffene Wahlentscheidung zu ermitteln, sobald das Verfahren für die verwendete Schlüssellänge gebrochen werden kann.

Daher listen wir in dieser Arbeit zunächst alle Angriffe auf das Wahlgeheimnis auf, denen ein verifizierbares Wahlsystem ausgesetzt sein kann. Im Anschluss daran werden Lösungen für einzelne Schwachstellen präsentiert. In dieser Arbeit haben wir uns dabei auf drei Fragestellungen konzentriert: (1) Wie kann die korrekte Anonymisierung von Stimmen mit einem generischen Verfahren gezeigt werden ohne dabei das Wahlgeheimnis zu verletzen? (2) Wie können Wahlsysteme, die die abgegebenen Stimmen anonymisieren oder homomorph zählen so verbessern werden,

dass sie eine dauerhafte Geheimhaltung der Stimme garantieren? (3) Wie kann das Vertrauen, welches ein_e Wähler_in in Wahlhelfer_innen und Wahlvorstände legen muss reduziert werden?

Elektronische Wahlsysteme verwenden meist so genannte Mix-Netzwerke, um die kodierten Stimmen zu anonymisieren. Während der Anonymisierung veröffentlicht das Mix-Netzwerk Daten, mit denen Wähler_innen und Beobachter_innen kontrollieren können, dass keine Stimme verändert wurde. Die in der Praxis verwendeten generischen Ansätze bieten jedoch keinen ausreichenden Schutz vor Angriffen auf das Wahlgeheimnis. Um Frage (1) zu beantworten betrachten wir daher in dieser Arbeit verschiedene Ansätze und stellen ein neues Protokoll vor, welches auf bestehenden Verfahren beruht und diese hinsichtlich Effizienz und Geheimhaltung verbessert.

Ein weiterer Nachteil von Wahlsystemen, die eine Anonymisierung der abgegebenen Stimmen vorsehen ist, dass alle Mix-Netzwerke asymmetrische Verschlüsselungsverfahren verwenden und somit das Wahlgeheimnis lediglich für eine begrenzte Zeit gewährleistet ist. Bezüglich Frage (2) stellen wir daher ein Verfahren vor, welches die abgegebenen Stimmen mit einem Commitment-Verfahren kodiert und somit eine informationstheoretische Sicherheit gewährleistet. Dies garantiert, dass ein_e Angreifer_in selbst bei beliebig großer Rechenleistung nicht in der Lage ist, die solcherart geschützten Daten zu decodieren. Wie Mix-Netzwerke erlaubt auch unser Verfahren die Korrektheit des Anonymisierungsprozesses zu verifizieren. Selbst unter der Annahme, dass alle Wahlhelfer_innen kooperieren, um das Ergebnis der Wahl zu verändern, kann die Integrität der abgegebenen Stimmen gewährleistet werden. Der von uns entwickelte Anonymisierungsprozess ist dem klassischen Mix-Netzwerk sehr ähnlich und kann dieses in vielen Wahlsystemen ersetzen. Dadurch können bestehende Systeme so angepasst werden, dass sie eine dauerhafte Geheimhaltung der abgegebenen Wahlentscheidung gewährleisten. Um dies zu verdeutlichen zeigen wir in dieser Arbeit wie unser Anonymisierungsprozess verwendet werden kann, um die Wahlsysteme Prêt à Voter und Split-Ballot zu verbessern. In beiden Fällen müssen lediglich kleine Änderungen an den aktuellen Verfahren vorgenommen werden.

Ein ähnlicher Ansatz wird auch für Wahlsysteme erarbeitet, welche vorsehen die abgegebenen Stimmen homomorph zu zählen. Am Beispiel des Wahlsystems Scratch & Vote zeigen wir, wie unsere Lösung im Rahmen einer Präsenzwahl eingesetzt werden kann. Auch hier sind lediglich kleine Änderungen an dem aktuellen Verfahren notwendig.

Bezüglich Frage (3) untersuchen wir den Ansatz so genannter nicht personifizierter Belege. In diesem Fall erhält ein_e Wähler_in einen Beleg, der die Stimme anderer Wähler_innen und nicht zwangsläufig die eigene Stimme enthält. Dadurch stellen die Belege keine Verbindung zwischen einem Wähler beziehungsweise einer

Wählerin und seiner beziehungsweise ihrer getroffenen Wahlentscheidung her und Wähler_innen müssen Wahlhelfer_innen nicht vertrauen, dass sie diese Verbindung geheim halten. Wir stellen eine elektronische Wahlurne vor, die nicht personifizierte Belege erzeugt. Dabei wird ein Anonymisierungsprozess verwendet, der dem des Mix-Netzwerkes sehr ähnlich ist. Die Erzeugung der Belege ist universell verifizierbar und unser System verbessert die bestehenden Ansätze bezüglich Geheimhaltung der Stimmen und Korrektheit des Wahlergebnisses.

Abschließend vergleichen wir die von uns verbesserten Wahlsysteme, arbeiten deren Vor- und Nachteile heraus, und stellen eine Liste mit möglichen künftigen Arbeiten vor.

Contents

1 Introduction

1.1 Motivation

During the last decades, the number of countries using electronic voting systems for parliamentary elections increased constantly. Computerized solutions have been deployed, for instance, in the United States, Estonia, Norway, and Belgium. In the United States, voting machines were introduced in 1892 by mechanical lever machines used in Lockport, New York [Jon03]. Several election authorities around the country followed with a large variety of electronic voting systems, for instance, optical scan systems, introduced in 1962 in Kern City, California, and *Direct Recording Electronic Voting Machines* (DREs), introduced in 1976 in Illinois. In Europe, Belgium became one of the first countries using electronic voting systems through the introduction of DRE based systems in 1991. Eight years later, in 1999, they were deployed on a large scale and during the Belgian federal election in June 2007 already 44% of the voters have cast their vote electronically [CP07]. Often named advantages of electronic voting systems are savings in costs and a faster and more accurate tallying process that needs less administrative effort.

In Germany, electronic voting systems for parliamentary elections were first deployed in 1999. Voting machines produced by the company Nedap were sporadically used on several election levels. After the federal elections in 2005, two complaints were submitted to the Federal Constitutional Court and a judgement followed on the 3rd of March 2009. According to the verdict the used voting machines did not meet the requirements of the German Constitution. The court deduced from Article 38 in conjunction with Article 20.1 and 20.2 that it must be possible for citizens to check the essential steps in the election act, including the accurate counting of votes [Fed09, 39(71)].

Although the so-called "public nature of elections" is just a requirement for German parliamentary elections, since 2004 research in the field of electronic voting focuses on verifiability. Voters who are able to check, for instance, the accuracy of the tallying procedure put more confidence in the correctness of the result. Thus,

while developing electronic voting systems, transparency of the election process is a very important design goal. So-called "end-to-end verifiable" voting systems enable a voter to verify all steps of an election, from the vote casting process to the determination of the election result. Since these schemes allow checking the correct counting of votes cast in all polling stations, they provide an even higher extent of voter verifiability compared to the traditional voting system [HDV12].

Nevertheless, transparency also increases the risk of attacks regarding voter privacy. Thus, the deployment of an electronic voting system must pass not only the hurdle of the public nature of elections, it must also be assured that the principle of secret suffrage is not violated. This means that no one else but the particular voter should know for whom he or she voted for.

This is not only of high importance for secret suffrage but also for free suffrage. Suppose each voter receives proof that his or her vote was tallied as intended. In this case voters can also prove to someone else whom they voted for, which exposes them to external influences such as vote-selling or coercion. Furthermore, voting would not be free if voters have to fear the disclosure of their voting decision on the day of the election or afterwards.

1.2 High-Level Explanation of our Contribution

This work focuses on the evaluation and improvement of verifiable electronic voting systems with respect to privacy. We concentrate on solutions implemented for polling stations because these systems do not force a radical departure from the vote casting process the voter is used to. They do not require the ownership of a specific device, like a personal computer or a mobile phone. Furthermore, privacy issues, like family voting can be neglected because voters cast their vote in a controlled environment. Nevertheless, a couple of weaknesses and improvements presented in this work can be transferred to remote voting systems as well. Furthermore, our efforts primarily concentrate on schemes that allow generating encoded votes by filling out paper ballots. This prevents that voters have to interact with machines and that malicious devices are able to learn their vote cast.

1.2.1 Privacy Weaknesses of Verifiable Voting Systems

The first electronic voting systems had a significant drawback compared to the traditional system. While a manual counting process can be observed, the correctness of an electronic tallying procedure is harder to verify, especially for the average voters that do not have expert knowledge. Thus, the electronic voting systems of the

first generation assumed that voters put their trust in the correct functionality of the provided hard- and software.

To improve the transparency of electronically supported elections, in 1994 Benaloh and Tuinstra [BT94] introduced a voting system that is not only *universally verifiable*, but also *receipt-free*. This means that during the vote casting process, the voters receive a receipt and/or some additional information allowing them to verify that the election outcome was computed correctly (universal verifiability). At the same time the scheme prevents that voters can use this receipt and/or data and the public information to prove to someone else whom they voted for (receipt-freeness). However, their approach does not protect the voters from malicious authorities, that are able to coerce a voter and prevent him or her from casting a vote. Thus, this idea was further elaborated and new remote and poll-site voting systems were built. In this work we concentrate on poll-site voting schemes and with respect to this direction important improvements were *SureVote* [Cha04] presented by Chaum and Neff's *MarkPledge* [Nef04], both introduced in 2004.

Neff's approach requires an interaction between the voter and an election authority. Therefore, this scheme has been further developed to *Direct Recording Electronic Voting Machines* (DREs) and remote voting systems, for instance, the scheme proposed by Moran and Naor [MN06], *VeryVote* [JRF09], and *EVIV* [JFR13]. There are further DRE based and remote voting schemes that use other approaches, for instance, the protocol by Cramer et al. [CGS97] and STAR-Vote [BBE+13]. However, all these schemes require an interaction between the voters and a voting machine. The voting schemes *Prêt à Voter* [CRS05, RBH+09], *Scratch & Vote* [AR06], *Punchscan* [PH06], and its successor *Scantegrity* [CEC+08, SFCC11] are based on Chaum's idea. The distinguishing feature of these systems is that voters can cast encoded votes by filling out and scanning special ballot papers. This has a significant advantage compared to DRE based schemes because if electronic devices record the candidates selected by the voters, malicious code and hardware can be used to violate voter privacy.

Prêt à Voter, Scratch & Vote, and Scantegrity differ with respect to the information printed on receipts and how receipt-freeness is ensured. By filling-out the Prêt à Voter ballot, the voter generates a vote that is encrypted with a homomorphic public key encryption scheme. The encrypted vote is cast by scanning the ballot and the voter receives a copy of the recorded information as receipt. Afterwards, the votes cast must be anonymized to break the link to the respective receipt before they can be decrypted and tallied. Scratch & Vote also uses homomorphic encryption and the same receipts, but the votes cast are tallied homomorphically and decrypted afterwards. Since only the election outcome is decrypted, no anonymization process

is needed to preserve voter privacy.

In contrast, the election protocol of Scantegrity is based on confirmation codes that are generated by an unconditionally binding commitment scheme. During the vote casting process, codes for each marked candidate are revealed and serve as receipt. The association between the codes and the candidates is stored in private tables and must be kept secret to preserve voter privacy.

For many of these proposals the provided level of privacy has been well studied. Thus, in Chapter 3, we first summarize all vulnerabilities that might violate voter privacy or receipt-freeness in verifiable electronic voting schemes[1]. In a second step, solutions for several identified weaknesses are introduced. More precisely, we first provide a generic verification procedure for mix-nets that provides efficiency and privacy. Second, we introduce everlasting privacy to mixing and homomorphic tallying based schemes. And, third, analyze and improve voting systems that provide non-personalized receipts.

1.2.2 Privacy Preserving Generic Verification Procedure of Correct Shuffling

While verifiability of Scratch & Vote follows from the publicly performed tallying procedure, the anonymization process used within Prêt à Voter and the correctness of the Scantegrity tables must be verifiable. Most electronic voting systems use a reencryption mix-net [PIK93] in combination with a proof of correct shuffling to anonymize votes. Although a series of zero-knowledge proofs has been proposed, for instance, by Groth [Gro10], Lipmaa and Zhang [LZ12], and Wikström [TW10], in practice generic verification procedures are preferred. A legal analysis, for instance, showed that the German principles inclines us towards verification procedures like *Randomized Party Checking* [DHR+11]. This is due to the fact that the mechanism of random sampling and checking are more intuitive and comprehensible to the public, and in principle any observer can contribute to the random audit checks. Especially *Randomized Party Checking* enjoys great popularity and is not only used within Prêt à Voter [XCH+10], but also in online voting systems such as *Civitas* [CCM08]. Even though the information published during the verification process allows an attacker to map each voter to a subset of votes cast, other approaches, like the cut-and-choose based approach proposed by Sako and Kilian [SK95], are prevented because of their poor performance. Thus, one question we address in our

[1]Note that we do not discuss privacy problems specific to one system, for instance, the invisible ink used for Scantegrity. Prior to the assignment of a certain voting system further evaluation is needed, especially with respect to other electoral principles.

work is how to develop a generic verification procedure that provides both efficiency and privacy.

Message anonymization using a publicly verifiable reencryption mix-net works as follows: Each voter $i \in [1, K]$ encrypts his or her vote s using a homomorphic and probabilistic public key cryptosystem. The resulting set of ciphertexts

$$\{u(i)\}_{i=1}^{K} = \{\mathsf{Enc}(s(i), t(i))\}_{i=1}^{K}$$

is the input to the mix-net, where $t(i)$ is a random value. Then, each mix performs the following operations on its input set:

1. Mix M reencrypts each ciphertext i by multiplying it with $\mathsf{Enc}(0, t'(i))$, where $t'(i)$ is a value chosen uniformly at random.

$$u'(i) = \mathsf{Enc}(s(i), t(i)) \cdot \mathsf{Enc}(0, t'(i)) = \mathsf{Enc}(s(i) + 0, t(i) + t'(i))$$

$$= \mathsf{Enc}(s(i), t(i) + t'(i)).$$

2. The set of reencrypted messages $U' = \{u'(i)\}_{i=1}^{K}$ is permuted and the resulting set of votes

$$U_1 = \pi(U')$$

is published.

3. M publicly proves in zero-knowledge fashion that the data has been processed correctly.

4. The mix sends its output U_1 to the next mix.

Mixing has been performed correctly if the reencryption $u'(\pi(i))$ of $u(i)$ encrypts the same vote s

$$\mathsf{Dec}(u'(\pi(i))) = \mathsf{Dec}(u(i)) = s.$$

Since a homomorphic encryption scheme is used, a simple and efficient approach to verify correct mixing is to build the product over the set of input and output ciphertexts and verify whether they encrypt the same value, i.e.,

$$\mathsf{Dec}(\prod_{i=1}^{K} u'(\pi(i))) \stackrel{?}{=} \mathsf{Dec}(\prod_{i=1}^{K} u(i)) = \sum_{i=1}^{K} s(i).$$

This approach is known as *Optimistic Mixing* [GZB+02]. However, by replacing two ciphertexts, one for the preferred candidate and one to balance out the product, a malicious mix can violate correctness without being detected. This can be prevented

by randomly assigning the input ciphertexts to L blocks and verifying the correct reencryption of each block using optimistic mixing. More precisely, the following steps are performed:

1. Each input element is assigned to one of the L blocks.

2. For each input block the mix has to show a corresponding output block that consists of all elements of the input block in reencrypted form.

3. The correct reencryption of each block is verified by optimistic mixing.

The probability of an undetected coercion now depends on the chance that both modified ciphertexts end up in the same block.

This approach was used in Norway in 2011 [Gov]. However, only if all mixes are honest no information about the votes cast by individuals is revealed. Otherwise, each encrypted vote can be assigned to a subset of output blocks and therefore to a subset of votes cast. Furthermore, the mix-net verification process used in Norway is vulnerable to an attack revealed by Khazaei et al. [KTW12]. To solve these issues, we propose to carry out two mixing and two verification steps for each mix. This allows carrying out the block assignment in a way that even if only one mix is honest an attacker cannot assign one input ciphertext to one output block. More precisely, in the first verification round the ciphertexts are assigned to an input block at random. Afterwards, the elements of each output block generated in the first round are assigned to input blocks such that each input block contains at least one ciphertext of each output block. This is possible if each block has almost the same size L and if L is chosen such that for K inputs $L^2 \geq K$. In addition, we compare this approach with other common generic verification procedures and show that our solution is compatible with respect to correctness and efficiency.

The voting system Scantegrity [CCC+09] also uses a verification process similar to *Randomized Partial Checking* to prove that the confirmation codes and candidates were correctly mapped during the tallying procedure. However, this scheme has a further vulnerability: verifiability is implemented by confirmation codes shown to the voters during the vote casting process and published on the bulletin board. Thus, by making a photo of the filled-out ballot while the codes are shown next to the candidates, the voter can generate a proof of his or her vote cast. Improving the scheme such that it prevents this attack enforces large changes to the entire voting system, from the ballot layout and receipt generation process up to the tallying procedure. Thus, the modifications needed to ensure receipt-freeness for Scantegrity are out of scope in this work.

1.2.3 Everlasting Privacy

In Prêt à Voter and Scratch & Vote the votes cast depend on computational assumptions since these schemes are based on homomorphic encryption and/or on encryption mixes[2].

The requirement of *everlasting privacy* dates back to 1988. In [Cha88] Chaum presents the first protocol for elections with unconditionally-secret ballots. However, his approach requires an interactive ballot issuing protocol between the voters and the authorities and does not address receipt-freeness. Several protocols followed, for instance, the remote voting protocol by Cramer et al. [CFSY96] that improves Chaums approach with respect to efficiency. However, the first poll-site voting scheme that provides receipt-freeness, universal verifiability, and everlasting privacy was presented by Moran and Naor in 2006 [MN06]. Their approach uses generic commitment schemes with homomorphic properties, but requires that the voter uses a DRE to cast his or her vote, allowing a malicious device to violate voter privacy.

This scheme was followed by two paper based approaches: *ThreeBallot* [Riv06] and *Bingo Voting* [BMQR07]. However, in [Str06] and [ACG07] the authors showed that the privacy of the ThreeBallot voting system can be violated by statistical attacks. Furthermore, regarding Bingo Voting, the integrity of the recorded voting decisions relies on a trusted random number generator. By comparison, in most end-to-end verifiable voting systems the correctness of the election result can be ensured, even if all authorities are malicious and collaborate.

Shortly after Bingo Voting, in 2007 Moran and Naor proposed a paper based voting system providing everlasting privacy called *Split-Ballot* [MN07, MN10]. This approach is strongly based on Punchscan, but has a very interesting extra twist. The voter votes by splitting his or her choice over two ballots, which are sent to two different authorities. Before the votes are decoded and counted, these two authorities anonymize the vote shares using a mixing procedure similar to the process carried out by mix-nets. Although they are able to determine the election outcome, they cannot reconstruct an individual vote without conspiring. So there is no single point of failure with respect to privacy.

In 2007, Essex et al. introduced *Aperio* [ECA10], a voting system based on Punchscan that can be used for elections held in environments with low computerized support. In their paper the authors show that a verifiable voting system can be obtained without the use of computers, scanners, internet access, and cryptography.

[2] This is also a vulnerability of Punchscan and Scantegrity since these schemes use unconditionally binding, but only computationally hiding commitment schemes.

Furthermore, Aperio provides (everlasting) privacy under the assumption that no data is leaked during the ballot preparation and printing process. However, the ballots are still manually counted and verified. Thus, in contrast to other verifiable voting systems, the tallying and verification process is less efficient and needs a lot of administrative effort. Furthermore, since no data is published for auditing, the ability to verify the correctness of the election outcome depends on whether a voter can attend the verification procedure.

In 2009, van de Graaf proposed to merge Prêt à Voter and Punchscan using the former's ballot layout, and the latter's bit commitment scheme [Gra09]. The resulting voting system does not only provide verifiability and correctness but also everlasting privacy. This was the first publication proposing to replace the unconditionally binding and computationally hiding commitment scheme used in the standard Punchscan and Scantegrity version with a computationally binding and unconditionally hiding scheme to obtain everlasting privacy.

In 2010, Essex. et al introduced *Eperio* [ECHA10] which is based on Aperio but uses a simpler ballot layout and cryptographic setting to enable a computerized tallying and verification process. Like Punchscan and Scantegrity, this scheme does not provide everlasting privacy since computationally hiding commitment schemes are used. However, as shown by van de Graaf in [Gra09], this can be improved by replacing the cryptographic primitives.

An advantage of Eperio in comparison to the solution presented by van de Graaf is that the tables used to link the encoded ballots to the candidates selected are less complex. This scheme only uses a small set of cryptographic primitives to simplify the verification procedure and improves the execution time. The tables are generated and audited without any cryptography. Commitments are solely used to commit to their setup to prevent subsequent modifications. Although Eperio is an interesting solution since the auditing process is easier to understand for the average voters, this approach has several drawbacks.

A verification procedure similar to cut-and-choose is used. To increase the probability of detecting a coercion, several tables have to be generated and verified. Although the verification process is easy to understand and perform, this is a lot of work even for elections with only 1000 voters and 5 tables. Thus, just as with systems using cryptography, tool support during the verification procedure is inescapable. Furthermore, the cut-and-choose based verification procedure provides correctness with high assurance and only after several rounds. For mixing based schemes zero-knowledge arguments of correct shuffling can be applied which provide a stronger guarantee that the data was processed correctly. Furthermore, this scheme does not allow for homomorphic tallying which would reduce the compu-

tational and organizational effort for simple elections. Therefore, in this work we discuss how everlasting privacy can be introduced to mixing and homomorphic tallying based schemes.

Universally Verifiable Mix-Net Providing Everlasting Privacy

One solution that addresses everlasting privacy for universally verifiable mix-nets is Split-Ballot. However, the anonymisation protocol of Split-Ballot is devised for a very particular situation: two voting authorities which want to mix ballot shares of a specific format, allowing them to jointly reveal the selected candidates and compute the tally. In addition, their anonymization procedure makes intrinsic use of a very specific, numerical representation of the ballot shares, and it is neither obvious how to use their mixing protocol in other voting protocols, nor how to obtain a general, unconditional mix-net for mixing messages of arbitrary format. Despite the vast literature on mix-nets, we are not aware of any publication that accomplishes everlasting privacy for mixing. Thus, in this work we show how messages can be anonymized providing verifiability, but at the same time everlasting privacy.

A universally verifiable mix-net publishes information that allows observers and voters to check that the encrypted votes were processed correctly. Usually this audit information consists of the votes cast in encrypted form, the output of each mix, and the proofs of correct mixing. In standard mix-nets, privacy of this data is computational and anonymity of the sender is only guaranteed as long as the cryptographic assumptions last.

Our solution to this issue is by using a homomorphic and unconditionally hiding commitment scheme to encode the audit information. These commitments are computationally binding, allowing that a *prover* can commit to a certain value without being able to change his or her mind later on. Furthermore, they provide everlasting privacy because an unconditionally hiding commitment can be a commitment to any value of the message space with the same probability. One might argue that the drawback of unconditionally hiding commitment schemes is that they provide computational bindingness only. It follows that if the computational problem can be solved for the key length used, the election result can be changed in case of a recount. However, the integrity of the votes cast is guaranteed at the moment of vote casting and tallying. Furthermore, an unlimited repetition of the ascertainment of the results is not possible in the traditional voting system either. Till when the election outcome can be recounted depends on the period of safekeeping the election documents. In Germany, for instance, according to §90.3 Federal Electoral Code, paper ballots can be destroyed 60 days prior to the next elections. Against this

background, it is not necessary to require everlasting integrity.

In our solution the system commits to a vote cast s with a randomly chosen decommitment value t_0. Like with homomorphic encryption, each mix M can recode (or rerandomize) a commitment $u = \mathsf{Com}(s, t_0)$ by multiplying it with a commitment to a rerandomization value t_1, that is

$$u' = u \cdot \mathsf{Com}(0, t_1) = \mathsf{Com}(s, t_0) \cdot \mathsf{Com}(0, t_1) = \mathsf{Com}(s, t_0 + t_1).$$

Like for homomorphic encryption, recoding is possible without knowledge of the votes encoded. Furthermore, due to the homomorphic property of the commitment scheme the proof of correct shuffling is similar to the one used for standard reencryption mix-nets.

However, when unconditionally hiding commitments are mixed a problem occurs. To open them, one needs to know what the decommitment values $t_0 + t_1$ are. Unlike conventional encryptions, in which the votes can be uniquely determined, a rerandomized u' can represent any message. Its interpretation depends on the decommitment value.

Our solution is to send those decommitment values together with the vote s as auxiliary information through a **private** mix-net to which the public has no access. Any rerandomization $u' = u \cdot \mathsf{Com}(0, t_1)$ has matching reencryptions

$$\langle v', w' \rangle = \langle v \cdot \mathsf{Enc}(0), w \cdot \mathsf{Enc}(t_1) \rangle,$$

where $v = \mathsf{Enc}(s)$, $w = \mathsf{Enc}(t_0)$, and Enc is a suitable homomorphic encryption scheme.

Essentially we use two tightly synchronized mix-nets run by the same mixes: one which mixes commitments and is fully public, and a second mix-net to which the public has no access to transport the opening values by processing the homomorphic encryptions. Observe that the permutations and rerandomization values used in the private mix-net must be identical to those used in public mix-net to be able to open the commitments afterwards. Then, after the last mix M_n has published its data, v_n and w_n are jointly decrypted, yielding s and $t^* = t_0 + t_1 + \cdots + t_n$, the opening values of $u_n = \mathsf{Com}(s, t_0 + t_1 + \cdots + t_n)$.

The "simple" mix-net sketched in the above paragraphs already provides, verifiability, correctness, robustness, and everlasting privacy towards observers. No data published during the verification process reveals information which would enable a computationally unbounded attacker to decrypt the votes or to map an encrypted value to a decrypted vote. But it has one drawback: the first mix gets to see $\mathsf{Enc}(s)$. So when the encryption scheme gets broken, this mix, if dishonest, could reconstruct

the vote associated to a receipt. Thus, in addition we propose a more "complex" protocol where the votes are split in two (or more) parts, and are submitted to separate mix-nets. After the vote shares were anonymized, they are recombined, decrypted, decoded, and published by a special publication authority.

Verifiable Mixing-Based Voting Systems Providing Everlasting Privacy

The mix-nets presented in this work can be used to introduce everlasting privacy to existing voting protocols by replacing the encoding scheme and reencryption mix-net. We show this using the example of Prêt à Voter and Split-Ballot. By adapting Prêt à Voter such that it uses the simple mix-net the system can be upgraded to provide everlasting privacy. However, since the authorities see the encrypted opening values, certain organizational measures are needed to ensure everlasting privacy towards them. In addition we show how the mix-net protocol that uses secret sharing can be used together with Split-Ballot. Although the standard version of Split-Ballot already provides everlasting privacy, two malicious authorities are enough to reveal the association between a receipt and the respective vote cast. Our version improves the scheme with respect to computational privacy. It allows scaling the amount of malicious authorities needed to violate voter privacy as long as the computational assumptions hold.

Homomorphic Tallying Schemes Providing Everlasting Privacy

For homomorphic tallying based schemes the same approach as for mix-nets can be used. We replace the data published for auditing with an unconditionally hiding and additively homomorphic commitment scheme. A vote for candidate i is represented as a vector $\langle s_1, ..., s_m \rangle$ which is 0 everywhere, except in the ith position, where it equals 1. Each entry of this vector is encoded using a commitment scheme $u_i = \mathsf{Com}(s_i, t_i)$, while the corresponding opening values are encrypted to $v_i = \mathsf{Enc}(s_i)$ and $w_i = \mathsf{Enc}(t_i)$.

The total for candidate i can by computed by $u_i^* = \prod_j u_i(j)$, where the j in parenthesis denotes an index ranging over all voters. Because of the homomorphic property, we obtain

$$u_i^* = \prod_j \mathsf{Com}(s_i(j), t_i(j)) = \mathsf{Com}(\sum_j s_i(j), \sum_j t_i(j)) = \mathsf{Com}(s_i^*, t_i^*).$$

To publish s_i^*, that is the total of votes candidate i received, and show its correctness, the voting system needs to reveal s_i^* and t_i^*. This is possible because it knows the

homomorphically encrypted values s_i and t_i. By computing

$$v_i^* = \prod_j \mathsf{Enc}(s_i(j)) = \mathsf{Enc}(\sum_j s_i(j)) = \mathsf{Enc}(s_i^*)$$

and

$$w_i^* = \prod_j \mathsf{Enc}(t_i(j)) = \mathsf{Enc}(\sum_j t_i(j)) = \mathsf{Enc}(t_i^*)$$

and decrypting the results, the voting system finds s_i^* and t_i^*, respectively. To determine the tally, this procedure is applied to each candidate. Opening the commitment $u_i^* = \mathsf{Com}(s_i^*, t_i^*)$ by publishing the values s_i^* and t_i^* is sufficient to show correctness and to provide universal verifiability towards the public.

To some extend this protocol can be seen as a simplification of the remote voting system proposed by Cramer et al. [CFSY96]. However, their approach is designed for a multiple-authority election and uses secret sharing, while our protocol can also be run with only one authority. This allowed us to improve the remote voting system Helios with unconditionally privacy, as presented in [DGA12]. Furthermore, we show how our process can be applied to homomorphic tallying based poll-site voting systems using the example of Scratch & Vote. The homomorphic counters used in the classic scheme are encoded with an unconditionally hiding and homomorphic commitment scheme. By multiplying the commitments the votes per candidate are incremented. The opening values are privately processed and published at the end of the tallying procedure to prove correctness of the election result.

1.2.4 Towards a Voting Scheme Providing Non-Personalized Receipts

In 2007, Araújo et al. [ACG07] and Rivest et al. [RS07] introduced a different approach to issue receipts. This is, instead of receiving a receipt of his or her own vote, each voter gets one or more receipts that correspond to previously cast votes. To achieve this, the scheme uses a special ballot box that is publicly initialized with a certain number of votes prior the election (i.e., the initial votes). The tally is computed by subtracting the initial votes from the votes cast. While Rivest et al. discuss this type of receipt more as a possible enhancement of existing voting schemes, in [AR08] Araujo et al. develop a verifiable voting system that is based on the Farnel scheme [Cus01].

Non-personalized receipts have several advantages regarding privacy. Since voters do not receive a receipt of their own vote, they are not linked to their cast voting decision. Consequently, they do not have to put their trust in key holders and

authorities participating in the anonymization and tallying process. Furthermore, this approach has several advantages regarding verifiability. Usually the voters check whether their vote is included in the input batch of the tally by themselves. This allows attacks where malicious parties selectively replace votes of voters that most likely do not carry out this verification process, such as elderly or disabled people. Benaloh and Lazarus, for instance, described in [BL11] this weakness of verifiable electronic voting schemes as *Trash Attack*. But if each receipt contains a subset of votes cast, each vote may be verified by several people. Furthermore, the receipts are non-personalized and do not pose a risk to voter privacy; not even in the presence of distrusted authorities. Thus, voters that do not want to carry out the verification process by themselves can pass their receipt to helping organizations that check the correctness of the tallying procedure.

However, this approach also has some drawbacks. Each time a voter casts a vote in the special ballot box, a receipt is generated by spinning the box and scanning a subset of its content. This is time consuming and raises the amount of organizational work. Furthermore, using this simple approach, the probability of a vote being printed on a receipt depends on the time when the vote was cast, which allows attacks regarding privacy and correctness. By collecting some receipts an attacker can map a voter to a subset of votes cast. All votes that appear on receipts before the voter submitted his or her vote can be ruled out. Furthermore, the order in which votes are cast also determines the probability of each vote being handed out as a receipt and being verified by other voters. Thus, in this work we first introduce an electronic ballot box that efficiently generate receipts and second improve the receipt generation procedure with respect to correctness and privacy.

Non-personalized receipts can be generated electronically by a process that is similar to the anonymization procedure of mix-nets. All votes cast are reencrypted and mixed with a random permutation before they are printed on the receipt. To show correctness of this procedure standard proofs of correct shuffling can be applied. However, it would be very inefficient to carry out a shuffling and verification process each time a voter cast a vote. Thus, the device "preshuffles" all possible voting decisions during the setup process and marks the votes cast by setting flags. In addition, all filled out ballots are collected in a conventional ballot box. This allows verifying the correctness of the flags set by the electronic ballot box, i.e., if all votes cast were marked.

After the electronic ballot box holds the proofs and spot tests, all data stored is erased. Under the assumption that the secret permutation and the private values used to reencrypt the votes have been irrevocably deleted, the link between a vote cast and its anonymized version, printed on the receipts, is destroyed. Thus,

the output of the electronic ballot box, the set of anonymized votes cast, can be decrypted without violating voter privacy. Afterwards, the votes cast during the initialization are subtracted and the remaining votes are tallied.

In standard schemes the probability of a vote being printed on a receipt (and verified by a voter) depends on the time when the vote was cast. Therefore, we introduce weighted random selection to the receipt generation process carried out by the electronic ballot box. All votes cast receive a weight and for each vote printed on a receipt its weight is reduced decreasing its probability to be selected for the next one. To evaluate the impact of weighted random selection on privacy, correctness, and efficiency, we implemented the electronic ballot box and simulated the receipt generation process for several parameters. The results show that our proposal is efficient and improves existing solutions with respect to correctness and privacy.

1.2.5 Structure

The structure of this work is as follows: In Chapter 2 the security requirements and cryptographic primitives are introduced. In Chapter 3 the privacy weaknesses of poll-site voting schemes are summarized. Building on this, in Chapter 4 we describe a generic verification procedure of correct shuffling that provides privacy and efficiency. Afterwards, in Chapter 5, we introduce two universally verifiable mix-nets that provide everlasting privacy and in Chapter 6 apply them to the voting systems Prêt à Voter and Split-Ballot. In Chapter 7, we describe a homomorphic tallying protocol that provides universal verifiability and everlasting privacy, and we show how this tallying procedure can be used to improve the voting system Scratch & Vote. Finally, in Chapter 8 we analyze and improve voting systems that provide non-personalized receipts, using the Farnel voting system as an example. We conclude in Chapter 9 with a comparison of all improved voting systems and open issues that are left for future work.

2 Preliminaries

In this chapter we describe the security requirements and cryptographic primitives which are particularly important for our work.

2.1 Security Requirements

In the literature on electronic voting a vast amount of security requirements has been determined. While some definitions are established (e.g., verifiability), the specification of others (e.g., privacy) differs between several publications. Thus, in this section we review existing literature and define the requirements addressed in this work.

More precisely, we evaluate and improve voting systems with respect to *privacy* and analyze whether our proposals provide *verifiability*, *correctness*, and *robustness*. In this work we only look at poll-site voting schemes. Thus, we do not address properties like *democracy*, *fairness*, and *coercion-resistance* that are ensured by the organizational measures for polling stations defined by the German legal regulation. Furthermore, in this work we propose no substantial modifications of voting systems with respect to the ballot layout and vote casting process. Therefore, we do not consider properties like *usability*, *election versatility*, and *accessibility*.

2.1.1 Privacy Requirements

In our work we address the privacy requirements *voter privacy*, with respect to *computational* and *everlasting privacy*, and *receipt-freeness*.

A voting system provides **voter privacy** if it keeps the association between voters and their vote secret during all steps of an election, namely the election setup, vote casting, and tallying process. The latter election step also includes the verification procedure of end-to-end verifiable voting systems.

To analyze our schemes, we will use the following definition by Moran and Naor [MN06]. Voter privacy is preserved if an attacker cannot gain any information

about the votes cast by individuals apart from the final tally[3]. A voting system where voter privacy depends on computational assumptions is said to offer **computational privacy**. If even a computationally unbounded attacker cannot gain any information that helps him or her to violate voter privacy, the voting system provides **everlasting privacy**.

We define **receipt-freeness** as introduced by Benaloh and Tuinstra in [BT94]. An electronic voting system is *receipt-free* if either voters cannot create a proof of the content of their vote cast, or they are able to generate both proofs for real (tallied) and fake (not tallied) votes indistinguishable from each other. In a traditional paper based voting system, for instance, a voter could generate a proof by taking a picture of the filled out ballot paper. However, an attacker cannot distinguish whether this ballot has been cast or discarded. This holds true for passive attackers, as well as active attackers that coerce a voter prior to the vote casting process and force him or her to follow his or her instructions. However, also in the traditional voting system receipt-freeness has its restrictions. If the voter films the entire vote casting process, he or she can prove his or her voting decision towards an attacker. Thus, we concentrate here on providing the same level of receipt-freeness ensured by a traditional paper based voting system.

2.1.2 Correctness

A voting system provides *correctness* (or *integrity* [RBH+09]) of the election result if each valid vote is counted (completeness) and only valid votes are counted (soundness) [Pie08]. In other words it must be ensured that the vote generation, casting, and tallying procedure are carried out correctly without changing the voting decisions as they were intended by the voters. This is provided if votes are *cast as intended*, *recorded as cast*, and *counted as recorded*. In our work we further subdivide *cast as intended* in *encoded as intended* and *cast as encoded*. To prevent that a machine is able to learn how a voter voted, in many verifiable poll-site voting schemes the voter generates an encoded vote by filling out a special ballot paper that is scanned afterwards. In this case it is necessary to evaluate the correctness of the ballot and the scanning process separately. Therefore, such a more fine-graded distinction is appropriate. This leads to our definition of correctness, which is guaranteed if the following four requirements are fulfilled.

Encoded as Intended The filled out ballot reflects the voting decision intended by the voter.

[3]Note that it is impossible to prevent that if a candidate did not receive any vote, then a coercer can conclude that all voters did not cast a vote for him or her.

Cast as Encoded The voting system records (i.e., scans) the ballot correctly.

Recorded as Cast The recorded vote is stored and becomes input to the tally without being modified.

Counted as Recorded All stored votes are tallied correctly.

2.1.3 Verifiability

Verifiability is closely related to *correctness* and describes the ability to verify the correctness of the election result and especially the correct processing of votes. With respect to the scope of the provided auditing procedure a distinction is made between *individual* and *universal verifiability* [RBH+09].

Individual Verifiability A voting system provides individual verifiability if the voter can verify that his or her vote is *encoded as intended*, *cast as encoded*, and *recorded as cast*.

Universal Verifiability Universal verifiability is provided if all voters and observers can verify that all recorded votes are *tallied as recorded*.

End-to-End Verifiability So-called end-to-end verifiable voting systems provide both individual and universal verifiability.

2.1.4 Robustness

An electronic voting system provides *robustness* (or *resilience*) if an election can be run successfully, also in the presence of random faults and attempts to disrupt the election processes [RBH+09]. Thus, this property consists of two aspects. First, the ability to detect and resolve attempts to cheat and, second, the ability to run an election even in the presence of a minority of dishonest election authorities.

2.2 Cryptographic Primitives

The cryptographic primitives used in this work are homomorphic encryptions, commitment schemes, and universally verifiable mix-nets.

2.2.1 Homomorphic Encryption Scheme

A homomorphic encryption scheme is defined by a triple of algorithm $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that

1. $\mathsf{Gen}(1^\kappa)$ generates two separate keys, a public key pk and a private key sk, with respect to a security parameter κ. The private key is presumed to be shared among a set of key trusteed using *threshold decryption* (see, for instance, [GJKR99, Ped91] for ElGamal and [FS01, DK01] for Paillier).

2. $\mathsf{Enc}(s, r) = v$ denotes the encryption of message $s \in G$ with randomness $r \in H$ and public key pk.

3. $\mathsf{Dec}(v) = s$ denotes the decryption of ciphertext v to message $s \in G$ using private key sk.

4. The algorithm provides semantic security.

5. The algorithm is homomorphic in s and in r, meaning that for all $s, s' \in G$ and for all $r, r' \in H$:

$$\mathsf{Enc}(s, r) \cdot \mathsf{Enc}(s', r') = \mathsf{Enc}(s +_G s', r +_H r'),$$

   where $+_G$ and $+_H$ are operations in group G and H. As a consequence, reencrypting a ciphertext $v = \mathsf{Enc}(s, r)$ without changing the message s is possible by multiplying it with an encryption of the neutral element 0_G of group G, i.e.,

$$\mathsf{ReEnc}(v, r') = v \cdot \mathsf{Enc}(0_G, r') = \mathsf{Enc}(s, r) \cdot \mathsf{Enc}(0_G, r') = \mathsf{Enc}(s, r + r').$$

   Observe that reencryption is possible without knowledge of the message s.

For legibility, we will interpret a batch of votes from different voters as a vector, denoted by a capital letter. For instance, the batch of plaintext votes submitted is represented as $S = \langle s(1), \ldots, s(K) \rangle = \langle s(i) \rangle_{i=1}^K$. Operations on the entries of vectors carry over to the vectors, so we can write

$$V = \mathsf{Enc}(S, R) = \langle \mathsf{Enc}(s(i), r(i)) \rangle_{i=1}^K \text{ and}$$

$$\mathsf{ReEnc}(V, R') = \langle \mathsf{ReEnc}(v(i), r'(i)) \rangle_{i=1}^K = \langle \mathsf{Enc}(s(i), r(i)) \cdot \mathsf{Enc}(0_G, r'(i)) \rangle_{i=1}^K.$$

In this case it is implicitly understood that the $r(i)$ and $r'(i)$ are elements chosen uniformly random from the appropriate set. The Perm operation permutes the entries of a vector:

$$S' = \mathsf{Perm}_\pi(S) = \mathsf{Perm}_\pi(\langle s(i) \rangle_{i=1}^K) = \langle s(\pi(i)) \rangle_{i=1}^K.$$

2.2.2 Homomorphic Commitment Scheme

A (non-interactive) **commitment scheme** is a triple $(\mathsf{GenCom}, \mathsf{Com}, \mathsf{Unv})$ such that

1. $\mathsf{GenCom}(1^\kappa)$ generates the public commitment key ck for security parameter κ. Note that the security parameter defines the message space $\mathcal{M}$ and the randomization space $\mathcal{R}$. We will suppose implicitly the presence of ck in the remainder, leaving it out of the notation.

2. $u = \mathsf{Com}(s, t) \in \mathcal{C}$ takes as input a message $s \in \mathcal{M}$ and a uniformly chosen decommitment value $t \in \mathcal{R}$, resulting in a commitment $u \in \mathcal{C}$.

3. $\mathsf{Unv}(u, s, t)$ returns s if $u = \mathsf{Com}(s, t)$ and $\perp$ if not.

The commitment scheme has to provide the following properties:

Correctness For any $s \in \mathcal{M}, t \in \mathcal{R}$: $\mathsf{Unv}(\mathsf{Com}(s, t), s, t) = s$.

Non-Interactive All communication goes from the sender to the receiver.

Computationally Binding Given a commitment $u = \mathsf{Com}(s, t)$, for any PPT $\mathcal{A}$ the probability to find a second opening pair (s', t') with $s \neq s'$ such that $\mathsf{Com}(s, t) = \mathsf{Com}(s', t')$ is negligible in the defined security parameter κ.

Unconditionally Hiding For any pair $s, s' \in \mathcal{M}$ the distribution of the randomized values $\mathsf{Com}(s, t)$ and $\mathsf{Com}(s', t')$ must be identical when $t, t' \in \mathcal{R}$ are chosen uniformly random[4].

Homomorphic For all $s, s' \in \mathcal{M}$ and $t, t' \in \mathcal{R}$

$$\mathsf{Com}(s, t) \cdot_{\mathcal{C}} \mathsf{Com}(s', t') = \mathsf{Com}(s +_{\mathcal{M}} s', t +_{\mathcal{R}} t').$$

In the following, we will denote the neutral element $0_{\mathcal{M}}$ of $\mathcal{M}$ and $0_{\mathcal{R}}$ of $\mathcal{R}$ as 0. Furthermore, we will refer to the group operations $+_{\mathcal{M}}$ and $+_{\mathcal{R}}$ by $+$.

2.2.3 Universally Verifiable Mix-Nets

Mix-nets were introduced by David Chaum in 1981 [Cha81] to allow anonymous communication within a network. Its basic functionality is to process a set of input messages, so that any link between a single input and its associated output is

[4]This property can be weakened to statistically hiding, but for ease of exposition we do not explore this.

removed while the content remains unchanged. In the context of electronic voting, mix-nets are used to anonymize cast votes before they are decrypted and tallied.

Chaum's original approach is the so-called decryption mix-net, which makes use of public key cryptography. The reencryption mix-net introduced in 1993 by Park et al. [PIK93] is based on reencryption and takes advantage of the homomorphic properties of some public key encryption schemes. An important feature of this type of mix-nets is that its correctness can be verified by any third party, i.e., it can be shown by publishing additional audit information that each message that comes in, goes out.

Thus, an additional ingredient of reencryption mix-nets are **proofs of correct shuffling**. Each mix has to prove, in zero-knowledge fashion, that the data has been processed correctly, such that the set of output values is a valid shuffle of the set of input values. These proofs are made public, thus providing universal verifiability.

After the introduction of reencryption mix-nets in 1993 by Park et al., a large variety of shuffling proofs have been proposed, leading from generic verification procedures, like *Randomized Partial Checking* [JJR02] to zero-knowledge arguments of correct shuffling, like proposed by Groth [Gro10] and Lipmaa and Zhang [LZ12].

3 Privacy Weaknesses of Verifiable Poll-Site Voting Schemes

In this chapter we summarize privacy weaknesses of verifiable poll-site voting schemes. We do so with respect to voter privacy, in terms of computational and everlasting privacy, and receipt-freeness. A definition of these properties can be found in Section 2.1.1. Such an analysis allows us to identify open issues in the field of voting security and is the bases of our improvements presented in Chapter 4 to 8. The content of this chapter is published in [DH12] and [DHR+11].

3.1 Voter Privacy

Voter privacy is a very important requirement to guarantee a secret and free election. If voters fear that their voting decision may become public, they cannot vote freely and without influence. Furthermore, an attacker might exploit this situation to bribe or coerce voters to cast a vote for his or her favored candidate.

In the following sections we will discuss vulnerabilities that may arise during the election setup, vote casting, and tallying process and violate computational privacy. Everlasting privacy is addressed separately in Section 3.3.

3.1.1 Possible Violation During the Election Setup and Vote Casting Process

In this section we describe potential weaknesses of electronic voting systems that may occur during the election setup and vote casting process. In these election phases especially the electronic equipment poses a risk to voter privacy. If an adversary successfully manipulates some devices, he or she might be able to collect enough data that allows him or her to link a voter to the vote cast.

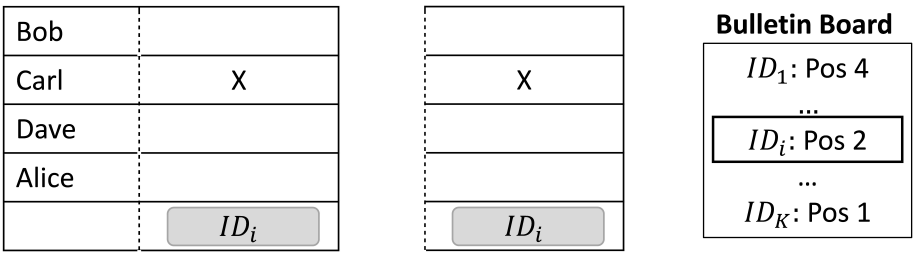


Figure 3.1: Ballot paper layout used in Prêt à Voter. Filled out ballot (left), scanned information (middle), and information published on the bulletin board (right).

Printer Knowledge: Some end-to-end verifiable voting systems propose the use of an adapted ballot paper which allows the voters to generate a receipt during the vote casting process. This receipt can be used to verify whether his or her own vote is included in the tally as cast.

In Prêt à Voter, for instance, the ballot paper consists of two halves which can be separated by a perforation down the middle. On the left hand side the candidates are listed in an arbitrary order. The right hand side contains a space against each name where the voters mark their choice and an ID that links to the secret candidate order in encrypted form (see Figure 3.1).

After the voter filled out the ballot, he or she detaches and destroys the human readable candidate list before leaving the secret polling booth. Subsequently, the right hand side is cast by scanning the ID and the marked positions. Each voter receive a receipt containing a copy of his or her recorded vote. Furthermore, the scanned information is published, for instance, on a publicly accessible bulletin board. Since all voters know the secret candidate order to their receipt, they can associate the marked positions with the respective candidates and verify that their own vote is included in the tally.

In this approach, voter privacy can be violated if an intruder manipulates the printer and reveals the association between candidate lists and IDs [RP10]. Since the positions marked on the ballots are published next to the IDs, an attacker can reconstruct for whom the owner of a receipt cast his or her vote. This weakness is also a vulnerability of other electronic voting systems where secret information is printed on ballots, such as Scantegrity and Scratch & Vote.

Information Leakage: When electronic devices are used, electromagnetic radiation and sometimes even transmissions from the inside of an electronic circuit can be obtained and analyzed [RP10]. Such electromagnetic emanations can, for instance,

come from the display or touch screen of a *Direct Recording Electronic Voting Machine* (DRE) [Kuh04, Sek10]. Important for the success of this attack is the strength of compromising radiation and to what extent this information can be isolated from other signals, like electronic noises [Kuh07]. In electronic voting, voters choose their candidates out of a limited set of voting options, which leads to a limited set of distinct signals and simplifies this attack.

The voting machines produced by the company Nedap, for instance, have been found prone to eavesdropping [GHB+06]. For devices of a special construction type signals that allow conclusions regarding the candidate selected can be received from a distance up to 25 meter. These signals are in the radio frequency and can be received using basic radio receiver equipment.

Electronic emanation is a vulnerability of many hardware components, for instance, screens, as shown for the Nedap machines, keyboards [VP09], and serial cables [Smu90]. Furthermore, the developers of a voting system should be aware of optical emanation caused by reflections in objects [BCD+09] and acoustic emanations caused by printers [BDG+10].

Storing Sensitive Information: Many electronic voting systems are designed to record votes electronically during the vote casting process. This may be achieved by a DRE or by scanning filled out ballot papers. In both cases voter privacy can be violated if the system additionally stores the sequential order of votes cast or the instant of time when a vote was submitted (accidentally, by poor design, or on purpose, by malicious software). If an adversary gets access to the machine, all he or she has to do is to observe the order in which voters cast their vote and allocate this information to the sequence of votes stored [KMHU04]. This vulnerability can occur in all electronic voting systems where the vote is recorded in clear (non-encrypted form) and in the presence of the voter.

Fingerprints: Many DREs list the candidate names in a constant order to offer identical ballots to all voters. If they provide a touchscreen or keys as user interface, fingerprints or fatty residues can reveal information about the candidates chosen by predecessors [Pie09]. This is a weakness of all electronic voting systems making use of an electronic device as user interface.

Information Added to Printouts: Sensitive data, like the instant of time when a vote is cast or the order in which votes were recorded can violate voter privacy. A malicious device that generates printouts can add such information by small dots, a modified ciphertexts, or an adapted barcode [KSW05] without being detected by

the voters. Collecting all printouts allows an attacker to get access to secret data without gaining physical access to the device. Such manipulations can occur in all electronic voting systems that record votes and generate printouts.

Single Point of Failure: As already pointed out by Moran and Naor [MN07], electronic voting systems should be developed in a way that they do not have a *single point of failure* with respect to security. If voters, for instance, use a DRE to fill out and cast their ballots, the machine and all authorities that have access to the device are able to reveal the votes cast. Thus, a malicious authority who observed the order or instant of time when the voters used the DRE is able to violate voter privacy. This is a vulnerability of all voting systems where the voters fill out and cast their votes using an electronic device.

3.1.2 Possible Violation During the Tallying Procedure

In this section we summarize privacy threats that may arise during the tallying procedure. In a traditional voting system the link between a voter and his or her vote gets broken at the moment when the vote is cast in the ballot box. However, this is not the case for verifiable voting systems that offer a receipt to the voter containing a link to his or her cast voting decision. Furthermore, the audit information that is published to show correct processing of votes may be a risk to voter privacy, too.

Authority Knowledge: As within a traditional voting system, it has to be ensured that privacy and functionality do not depend on only one person in authority. Thus, the private key to decrypt votes cast or to access secret information should not be handed out to one official only. This person could be bribed or threatened and compromising information can be revealed, for instance, by decrypting votes before the link to the voter has been removed.

Thus, so-called threshold cryptosystems are used. These schemes require two separate keys, one public and one private, whereas the private key consists of several key shares. In practice they could be stored on trusted devices, e.g., smartcards, and distributed among several persons in authority. As a result, a predefined number of officials has to be present to decrypt or access secret data. Nevertheless, regarding privacy, voters still have to trust that a threshold number of authorities act honestly and keep their key shares secret.

Another example for a shared responsibility are reencryption mix-nets, which are used in verifiable voting systems, such as Prêt à Voter, to anonymize votes cast. A reencryption mix-net consists of several *mixes* which are serially connected. The

first *mix*, for instance, a server controlled by a person, takes a batch of votes, changes the appearance by reencrypting the secret and by shuffling the order with a secret permutation. Afterwards, the output data is passed to the next mix which processes the votes the same way until the whole mixing process is completed. Each mix knows the association between its input and output batches. Therefore, the link between a voter and his or her vote cast is destroyed only when at least one mix is honest and keeps the permutation used secret. Summarized, all electronic voting systems using threshold-cryptography (e.g., Prêt à Voter and Scratch & Vote), using mix-nets (e.g., Prêt à Voter), and managing access to secret data (e.g., the tables used in Scantegrity) assume trust in a threshold number of officials.

Vote Clustering: Some voting systems publish data during the auditing procedure that allows to reveal information about the votes cast by individuals. This happens, for instance, if a polling station provides more than one electronic device that records votes and the outcome for each device is determined separately. In this case an intermediate result can be mapped to a specific device and consequently to its users. This leads to an additional association between a subgroup of voters and a subset of votes cast.

More precisely, imagine two scanners, S_1 and S_2, are used in a polling station. In this case the voters are grouped into two blocks: one block consists of the voters who cast their vote at scanner S_1 and the other block contains the voters who used scanner S_2. Assume just one of the two scanners recorded a vote for candidate "Alice", let's say S_1. In this case it is possible to say that a subset of voters, all that cast their vote at S_2, did not cast a vote for "Alice". Although the voting decision is not revealed, the assortment of possible candidates gets smaller. This is unacceptable because if a voter has been coerced into voting for "Alice", the coercer knows that he or she did not follow the instruction.

Such a fragmentation could also occur during the verification of a mix-net, depending on the verification method used. In [Cha02] Chaum describes this problem for mix-nets verified with Randomized Partial Checking [JJR02]. Summarized, vote clustering is a vulnerability of electronic voting systems where either: votes are cast and tallied at several devices; devices sign votes or receipts; or the votes are grouped during the verification procedure.

3.2 Receipt-Freeness

Verifiable electronic voting systems offer each voter the opportunity to revise all steps of an election regarding integrity. However, if receipts or audit information reveal secret data that allows conclusions regarding their vote cast, voters could be exposed to external pressures such as vote-selling or coercion.

Unique Ballots: A coercer can, for instance, force voters to take a picture of themselves and their filled out ballot paper. End-to-end verifiable voting systems offer a receipt during the vote casting process that is associated to a recorded ballot paper containing a unique ID. To allow individual verifiability, the voter can use this ID to verify whether his or her own vote is contained in the tally. However, this also enables the coercer to check whether the vote, shown on the photo, has been cast. Conversely, in the traditional voting system voters can ask for a new ballot since the ballot papers are not unique.

In Scantegrity, for each candidate marked by the voter, a unique confirmation code is revealed. To verify that the vote has been recorded as intended, the voter has to write down these codes and check on the bulletin board whether they appear. The codes disappear[5] before the ballot is scanned. However, if the voter takes a picture while they are still visible, for instance, in the secret polling booth, he or she can prove the association between a unique confirmation code and the corresponding candidate. Furthermore, if the codes appear on the bulletin board, the coercer can be sure that this ballot was cast.

To guarantee that voters cannot be bribed or coerced, they should not be able to generate a proof of their vote cast. Thus, verifiable voting systems may not bring the voting decision in clear words and the matching ID on one ballot. As the receipt used for verifying the own voting decision is handed out to the voter, the information printed on it cannot be classified as private. A coercer could force the voter to hand out the receipt or he or she shows it to another person voluntarily. Although it does not reveal any provable details about the voter (e.g., name or address), the possession of it signalizes to whom it belongs to.

[5]The codes and bubbles are printed with invisible ink. The color used for the confirmation codes has a shorter reaction time than the ink used for the bubbles. Thus, after marking a bubble, the codes turn black and are visible for some minutes. Afterwards, the bubbles turn black and hide the codes.

3.3 Everlasting Privacy

The requirement of vote secrecy is fulfilled if an electronic voting system keeps the association between a vote cast and the voter secret during all steps of an election. To that end, many voting systems make use of encryption to protect secret information, for instance, votes cast that are published for auditing. However, the security of an encryption scheme can be ensured just for a limited (and not assessable) duration of time. It is possible at any moment that an effective attack or technical innovations (e.g., quantum computers) allow breaking the underlying computational problem for the key length chosen. An exact period of validity cannot be determined but data encrypted with RSA and a key length of 2048 bit, for instance, could be disclosed in approximately 20 to 30 years [Kal03, FS03]. In addition, since storage is becoming cheaper every day, we must assume that the data on the bulletin board will be stored forever. This means that as soon as the the encryption scheme gets broken for the key length used, it will be possible to decrypt all published information.

In the case of Prêt à Voter, for instance, an arbitrary candidate order is shown on each ballot. During vote casting, the voter receives a receipt containing the positions marked and a link to the published candidate order in encrypted form. As a consequence, as soon as the cryptosystem is broken for the key length chosen, the attacker can reveal the associated candidates the owner of a receipt voted for. As already mentioned in Section 3.2, the possession of a receipt refers to a specific person what allows the attacker to violate voter privacy. The same applies to systems that provide universal verifiability by publishing some audit information in encrypted form. The overwhelming majority of these voting protocols is based on homomorphic encryption and/or on reencryption mixes, so what all these protocols have in common is that they provide computational privacy only.

In the context of electronic voting computational privacy is not sufficient. The disclosure of filled out ballots, even decades later, can still be highly embarrassing. For instance, think of a 65-year old presidential candidate whose voting behavior when he or she was 20 becomes public. It might ruin his or her chances. Also for elections that are not on state level voter privacy is of interest. If a university or company elects a new director it would be unpleasant for each candidate, also several years after the election, to detect which co-worker voted against him or her. Furthermore, computational privacy could increase the possibility for some nasty scenarios, for instance, a dictator who has come to power goes after people who have voted against him or her (or his or her ancestors) several decades ago.

Already in 1985 Chaum argued, in the context of credential mechanism [Cha85], that privacy should be everlasting, since individuals cannot be expected to assess

the strength of cryptographic mechanisms. Furthermore, a legal analysis of privacy weaknesses in poll-site eVoting systems showed that only if the secrecy of the vote is ensured for an indefinite period of time, a voting system is in accord with the principle of free suffrage [DH12]. Already the mere possibility of this type of disclosure could have a negative effect on the voting behavior because individuals might feel constrained about the content of their ballots submitted.

3.4 Evaluation of Identified Privacy Weaknesses

There are already several approaches to prevent the attacks described in Section 3.1.1. Ryan et al. [RP10], for instance, proposed a distributed ballot preparation and printing process to address **printer knowledge**. Regarding **information leakage**, it is possible to evaluate the emission prior to the election by an independent institute and implement security preventive measures according to the results. Also regarding the **single point of failure**, there are several proposals addressing this vulnerability. In Scratch & Vote, Prêt à Voter, and Split-Ballot, for instance, encrypted votes are generated by filling out paper ballots that are scanned afterwards. Together with a distributed printing process and threshold decryption, they avoid a single point of failure and ensure that electronic devices are not able to violate voter privacy. Since the scanners only record public and no secret information, this approach also prevents attacks based on **fingerprints**, **storing of sensitive information**, and **information added to printouts**.

As described in Section 3.1.2, there are mainly two vulnerabilities that can occur during the tallying process: vote clustering and authority knowledge. **Vote clustering** caused by the use of several technical devices can be prevented by publishing the overall outcome only. Furthermore, verification procedures that are vulnerable to this attack should not be used. However, with respect to generic verification procedures, the existing solutions either provide privacy and a poor efficiency, or they have a good efficiency but lead to vote clustering. Thus, further research is necessary to find an appropriate solution (see Chapter 4).

Although **authority knowledge** might be acceptable if a secret is shared among several persons in authority, systems with weak trust assumptions should be preferred. There are already some approaches, like Twin [RS07] and the verifiable version of the Farnel voting scheme [ACG10] that address this issue. However, before these schemes can be used, further research with respect to the provided level of correctness and privacy is needed (see Chapter 8).

As discussed in Section 3.2, problems with **unique ballots** generally arise when

voting systems use ballot papers marked with IDs. However, if the voter does not have to write down confirmation codes which are shown to him or her during the vote casting process, like proposed for Scantegrity, this problem is somewhat easy to solve.

A simple solution is to hide all information that allows identifying one single ballot paper, like the ID, by a scratch strip or invisible ink (like proposed for Prêt à Voter in [RP05]). Thus, a photo would prove how the voter filled out a ballot but not that this ballot has been cast. In this case additional organizational measures are needed to ensure that characteristics determining one specific vote are not disclosed as long as the ballot paper is under the ownership of the voter. The poll workers have to ensure, for instance, that the ID is not revealed before the ballot is scanned.

As shown in Section 3.3, many of the current schemes, like Prêt à Voter and Scratch & Vote do not provide **everlasting privacy**. The main problem is that all votes cast are publicly processed and that the voter receives a receipt containing his or her vote in encrypted form. Thus, for mixing and homomorphic tallying based schemes, verifiability processes should be designed such that all information published for auditing provide information-theoretical security (see Chapter 5, Chapter 6, and Chapter 7).

4 Privacy Preserving Generic Verification Procedure of Correct Shuffling

For parliamentary elections it is very important that voters and observers are convinced that the election outcome has been determined correctly. Therefore, one of the German principles, the public nature of elections, requires that all essential steps of the elections are subject to public observation [Fed09]. To solve this issue, several universally verifiable reencryption mix-nets have been proposed (see Section 2.2.3). However, as was shown in Section 3.1.2 it must be prevented that the verification procedure used leads to vote clustering. Thus, after an introduction to reencryption mix-nets, the existing generic verification procedures are evaluated, and an improved process, *Random Block Verification*, is proposed which is efficient and fulfills the requirement of voter privacy. The content of this chapter is published in [DJV12].

4.1 Reencryption Mix-nets

This section describes how standard reencryption mixing works. Furthermore, we give an overview of the assumptions made and the properties achieved. For more information see, for instance, [SK95] or [JJR02].

4.1.1 Parties Involved

The following parties are involved in the anonymization process:

Voters We have K voters that will submit a vote, for instance, by scanning a filled-out ballot, which is published in encrypted form. We will write the indexes i of the voters between parenthesis.

Mixes The mix-net consists of n mixing authorities called mixes, M_1,M_2,...,M_n.

Authorities The set of authorities consists of all private authorities, for instance, the key trustees, which participate in the reencryption mixing process.

Key Trustees The set of key trustees is a subset of authorities who hold shares of private or access keys, needed to decrypt data or access hardware.

Bulletin Board There exists a secure append-only public bulletin board, i.e., one to which only authorized entities can append, nothing can be deleted, and everyone has a consistent read access. This is a fairly standard assumption. See, for instance, [HL08] for more details.

Random Beacon We assume that all random challenge bits used in the verification steps come from a trusted beacon. This assumption is fairly standard. See, for instance, [MN10].

Auditors Auditors are experts who run the verification process, check the published proofs, and certify the output. Furthermore, they execute and verify the random beacon's generation of random bits.

Observers Observers are interested parties, such as voters, political parties, authorities, international observers, and third parties who verify the correctness of the mixing process.

4.1.2 Assumptions

For a reencryption mix-net to be secure the following assumptions have to be made:

Assumption M.1 The authorities cannot break the computational problem of the encryption scheme.

Assumption M.2 At least one mix is honest and keeps the association between its input and output values secret.

Assumption M.3 Using (k, n)-threshold decryption at least $(n-k+1)$ key trustees act honestly and keep their key share secret.

Assumption M.4 All random challenge bits used in the mixing and verification steps produced by the random beacon are unpredictable.

4.1.3 Reencryption Mixing Process

Reencryption mixing is based on a **homomorphic public key encryption algorithm** which allows reencryption of encrypted messages, as described in Section 2.2.1. The purpose of the mixing process is to reencrypt and shuffle the batch of encrypted votes, such that at the end all votes can be decrypted without being able to link any of them to a receipt handed out to a voter. Furthermore, each mix has to publicly show that all votes contained in the input set are part of the output.

In the following we will provide a detailed description of the mixing process. As defined in Section 2.2.1 the operation ReEnc reencrypts an encrypted vote $v(i) = \mathsf{Enc}(s(i), r(i))$ with a random value $r'(i)$:

$$\mathsf{ReEnc}(v(i), r'(i)) = \mathsf{Enc}(s(i), r(i)) \cdot \mathsf{Enc}(0_G, r'(i)) = \mathsf{Enc}(s(i), r(i) + r'(i))$$

and the operation Perm permutes the entries of a vector with a random permutation π:

$$\mathsf{Perm}_\pi(\langle v(i)\rangle_{i=1}^K) = \langle v(\pi(i))\rangle_{i=1}^K.$$

Phase I: Vote submission Suppose a set V of encrypted votes has been generated. The ciphertext $v(i) = \mathsf{Enc}(s(i), r(i))$ denotes an encryption of vote $s \in \mathcal{M}$, encoded with randomness r, cast by voter i.

All submitted ciphertext $v(i)$ are published on the bulletin board, together with the receipt identification i. The input batch for the first mix is defined as $V = \mathsf{Enc}(S, R)$.

Phase II: The mixing process Let V be M_1's input batch. Then M_1 chooses a random permutation π_1, reencrypts and shuffles V:

$$V_1 = \mathsf{Perm}_{\pi_1}(\mathsf{ReEnc}(V, R_1)),$$

where the random values R_1 are uniformly generated by M_1.

The output of mix M_1 is input to the next mix M_2 which processes the data the same way:

$$V_2 = \mathsf{Perm}_{\pi_2}(\mathsf{ReEnc}(V_1, R_2));$$

and so forth.

In addition each mix has to publicly prove towards the auditors and observers that it has knowledge of the reencryption values and permutation used.

Phase III: Decoding and publication of the messages After the last mix M_n processed the data and the correct functioning of each mix has been checked, the ciphertexts are decrypted with the help of the key trustees and the set of votes S is published.

Phase IV: Certification of the output The auditors verify whether all public proofs of knowledge hold and certify the output. Table 4.1 summarizes the mixing process.

Table 4.1: Summary of the mixing process

	input from users	Mix i, for $i \in [1, n]$	output	decoding
public	$V = \mathsf{Enc}(S)$	$V_i = \mathsf{Perm}_{\pi_i}(V_{i-1} \cdot \mathsf{Enc}(0))$	V_n	$S = \mathsf{Dec}(V_n)$

4.1.4 Properties

Under the assumptions listed in Section 4.1.2, reencryption mix-nets have the following properties:

Individual Verifiability All voters can convince themselves that their vote is included in the input batch.

Privacy The identity of the sender of each vote remains secret as long as Assumptions (M.1), (M.2), and (M.3) hold.

Correctness Even if all authorities conspire, they cannot change the contents of any vote without being detected with overwhelming probability, as long as Assumptions (M.1) and (M.4) hold.

Universal Verifiability Any interested observer can verify that the mixing process was performed honestly by checking the published proofs of correct mixing.

Robustness If all authorities follow the protocol correctly, then the protocol always terminates successfully.

4.2 Evaluation of Existing Generic Verification Procedures

In this section we give an introduction to existing generic verification procedures and analyze whether they provide voter privacy. In mix-nets, privacy is the question of

how traceable a given ciphertext is. More precisely, looking at one input ciphertext there are some output ciphertexts which may be its reencryption while there are others that can be ruled out. The size of the group containing all possible outputs, the "anonymity group" (AG), determines how much privacy is achieved by the mix-net. In general, as motivated in Section 3.1.2, we require that the size of the anonymity group is K for K input elements. Since mix-nets should provide voter privacy, even if only one authority is honest, we will determine the anonymity group for this case.

The generic verification procedures existing in the literature can be assigned to two basic directions: Optimistic Mixing [GZB+02] and Cut-and-Choose [SK95].

Optimistic Mixing The verification procedure proposed by Golle et al. [GZB+02] is the first *optimistic mixing* approach. The input of a reencryption mix-net needs to be encrypted using a homomorphic public key cryptosystem. So, the idea is to use this property to prove the correct shuffling for the whole set of input and output elements. After the mix-net processed the data, for each mix two products are computed, one by multiplying all input ciphertexts and one by multiplying all output ciphertexts. Subsequently, the mixes are asked to prove that both results encrypt the same message, i.e., sum or product of the encrypted votes.

The input of the mix-net proposed by Golle et al. consists of a double encrypted vote and a hash value over the single encrypted vote. The hash values are used to ensure that votes cannot be changed without being detected since input $\mathsf{Enc}(4) \cdot \mathsf{Enc}(6) = \mathsf{Enc}(4+6)$ and output $\mathsf{Enc}(3) \cdot \mathsf{Enc}(7) = \mathsf{Enc}(3+7)$ leads to the same encrypted message. If a manipulation is detected, the double encryption allows for "back-up" mixing. The outer layer can be revealed and the inner layer can be input to a slower and more fine-graded verification process, for instance, a zero-knowledge argument of correct shuffling. However, multiple flaws were identified by Wikström [Wik03] and Abe and Imai [AI03]. The use of hash values, for instance, allows to violate voter privacy in the presence of a malicious voter.

Proof of Subproduct The verification procedure *Proof of Subproduct* [BG02] proposed by Boneh et al. is an improved optimistic mixing based process that is resistent against the weaknesses shown for the classic approach. For the verification process, a security parameter $\alpha \leq 5$ is defined which declares the number of subsets (which in the following we will refer to as blocks) to be checked. Afterwards, α blocks are generated, each of size of $\frac{K}{2}$ for a total number of K ciphertexts. Each input ciphertext is included in block P_i independent at random with probability $\frac{1}{2}$. To demonstrate that the input was processed correctly, the mix has to show for each

block the corresponding output ciphertexts and prove that the product of both sets encrypt the same message.

A higher value for α results in a stronger guarantee of correct mixing but also offers less privacy because the votes are grouped in smaller blocks. The authors show that the average anonymity group size is $\frac{K}{2^\alpha}$. However, even for a small α this verification process divides the K inputs at least in two sets violating voter privacy. Furthermore, the probability to cheat, estimated by the authors, is still around $(\frac{5}{8})^\alpha$. Therefore, Boneh et al. recommend the use of a slower verification protocol in parallel to guarantee correctness.

Cut-and-Choose Another approach to show correct shuffling is to use a cut-and-choose based verification process, as described in [SK95]. After the mix-net anonymized the encrypted votes, each mix must show that it knows permutation π and rerandomization values R such that $V_2 = \mathsf{Perm}_\pi(\mathsf{ReEnc}(V_1, R))$ for input V_1 and output V_2. This is accomplished by breaking the permutation π in π_1 and π_2 and correspondingly split the rerandomization values R in R_1 and R_2 such that

$$V_2' = \mathsf{Perm}_{\pi_1}(\mathsf{ReEnc}(V_1, R_1)) \text{ and}$$

$$V_2 = \mathsf{Perm}_{\pi_2}(\mathsf{ReEnc}(V_2', R_2)).$$

The intermediate batch V_2' is published together with commitments to π_1 and π_2 on the bulletin board. To check correct mixing, the verifier publicly issues a challenge, Left or Right. If Left, then the mix opens π_1 and R_1, if right, then the mix opens π_2 and R_2. The verifier checks the correctness of the output and continues with the next mix. Malicious mixes are able to generate an intermediate batch but cannot open it to both sides since they must either compute π_1 and R_1 out of π_2, R_2, π, and R or the other way round.

It follows that a malicious mix still has a chance of $\frac{1}{2}$ to replace all encrypted votes without being detected. Therefore, this process needs to be repeated for different permutations, decommitment values, and challenges until the probability of a successful coercion is below a certain value, defined by a security parameter.

Randomized Partial Checking The primary drawback of the standard cut-and-choose verification process is that it is inefficient in terms of computation and communication. Furthermore, not only the auditors but also the observers, like voters, have to verify each round making this approach impractical, especially for large-scale elections.

Randomized Partial Checking reduces the communication and computational costs by generating only one additional set. The basic idea is that each mix shuffles

its input two times and after the mix-net has completed its computations, it shows for each element in the intermediate batch either its correct association to one input element or to one output element (e.g., decided by random coin) [JJR02].

However, this approach has two drawbacks. First, the probability of modifying a small amount of votes is still quite high. The chance that l replaced votes pass the verification process without being detected is $\frac{1}{2^l}$ because just the half of the associations is checked. Second, as already identified by Chaum in [Cha02], this approach does not provide voter privacy due to vote clustering. Depending on a coin flip, the verification procedure either reveals the link between an intermediate ciphertext and the input (left), or its link with an output ciphertext (right). Assuming, the coin is fair, 50% of the left associations is opened, and similarly 50% of the right associations. Hence, on the average $\frac{K}{2}$ "right" links are not revealed and must belong to the input ciphertexts whose "left" link was opened and checked. Thus, each vote is hidden in an anonymity group of size about $\frac{K}{2}$.

Furthermore, an analysis of Randomized Partial Checking carried out by Khazaei et al. [KW13] showed that this approach is vulnerable to several attacks against both privacy and correctness.

Norwegian Mix-Net Puiggalí et al. combined in [AC10] the advantages of Optimistic Mixing and Randomized Partial Checking to improve the existing schemes in terms of correctness while maintaining voter privacy. Their work was incorporated into the Norwegian Evote Project[Gov] and used for a limited number of municipality elections in Norway. The Norwegian Mix-Net works as follows:

1. An independent verifier chooses a permutation at random and applies it to the set of input votes.

2. The list of votes is divided into $\sqrt[n]{K}$ equally-sized blocks, where n denotes the number of mixes and K the number of input ciphertexts.

3. For each input block, the first mix shows the corresponding output block, containing all elements of the associated input block in reencrypted form. The mix needs knowledge of the permutation used to answers this challenge.

4. All elements of the input block and all elements of the corresponding output block are multiplied. To show correct shuffling, the mix proves that the product of the output block is a correct reencryption of the product of the input block using a zero-knowledge proof. Thus, any observer can verify that the sum or product of votes encrypted in both sets is equal.

5. The verifier checks the proofs published by the mix.

6. This process is carried out for each mix. Apart from the first mix, the assignment of input ciphertexts to blocks depends on the previous node's assignment ensuring an equal distribution of input ciphertexts over all final output blocks of the mix-net (similar to the improved Randomized Partial Checking proposed by Chaum [Cha02]).

To replace votes without being detected, a malicious mix has to modify two votes: one vote to the preferred candidate and a second vote so that the product remains the same. Furthermore, the mix has to hope that both ciphertexts end up in the same block. It follows that the probability of modifying two votes without being detected is $\frac{\sqrt[n]{K}-1}{K-1}$ (see [AC10] for details).

Regarding privacy, Puiggalí et al. state that each output block of the last mix node is composed of at least one ciphertext of each input block of the first mix node. However, optimal privacy in the Norwegian Mix-Net is only ensured if all mixes are honest. But this is not the idea of a mix-net where privacy should already be provided if at least one single mix is honest. In this case, the size of the anonymity group is only $\sqrt[n]{K}$. Furthermore, by carrying out a cryptoanalysis of this approach, Khazaei et al. identified several flaws [KTW12] which allow replacing some inputs without being detected and violate privacy of a few voters. Some of the threats result from the dependency between the output blocks of one mix and the assignment of the input blocks of its successor.

4.3 Random Block Verification

To prevent the flaws identified for the Norwegian Mix-Net each mix should be verified independently. Furthermore, to provide optimal voter privacy, an anonymity group of size K has to be provided. Consequently, the block fragmentation should not help to associate one output element of the mix-net to a subset of mix-net inputs, even if only one mix is honest. Thus, first, we propose to build single mixes similar to Randomized Partial Checking where each mix shuffles twice. Consequently, the verification process is split in two steps: the correct reencryption of the input batch and the correct reencryption of the intermediate batch. We will refer to this as the "first" and "second" shuffling step. This allows us to build the blocks in the second shuffling step in a way that already for one mix an anonymity group of size K is provided. Second, we require that each ciphertext is assigned to exactly one block to prevent overlapping of subsets.

Notation Using Random Block Verification (RBV) the mix-net has to shuffle its set of input values $V = \{\langle v(i) \rangle\}_{i=1}^{K}$ two times. The output of the first mix M_1 is V' after the first and V_1 after the second shuffling step. The output of M_1 is the input to mix M_2 which correspondingly outputs V_1' after the first and V_2 after the second shuffling step.

Verification Process Note that this verification process takes place after each mix has shuffled its input and published its output.

1. After each mix has completed its computations, the set of input ciphertexts is permuted
$$V^* = \mathsf{Perm}_{\sigma_1}(V),$$
using a randomly chosen permutation σ_1.

2. The set of permuted input ciphertexts V^* is divided into $L := \lfloor \sqrt{K} \rfloor$ blocks denoted as $P_1^{In}, \ldots, P_L^{In}$. It follows that we have $R := K - L^2$ blocks of $L + 1$ elements and $L - R$ blocks of L elements (see Figure 4.1).

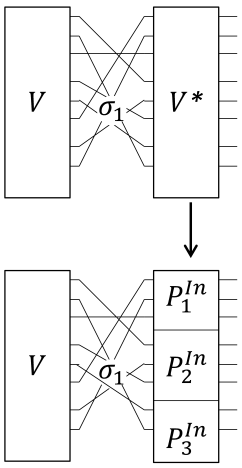


Figure 4.1: Verification procedure for 8 votes after Step 2

3. For each block P_l^{In} with $l \in [1, L]$, mix M_1 has to show the corresponding output block P_l^{Out} that is composed of the ciphertexts contained in the input block in reencrypted form (see Figure 4.2). As a consequence, the product over the elements contained in the output block encrypts the same value than the product over the elements contained in the input block. More precisely, assume the input block P_l^{In} contains three ciphertexts $u_1 = \mathsf{Enc}(s_1, r_1)$, $u_2 =$

$\mathsf{Enc}(s_2, r_2)$, and $u_3 = \mathsf{Enc}(s_3, r_3)$ encrypting message s_1, s_2, and s_3. Then the output block P_l^{Out}, if generated correctly, contains the same three messages, permuted using permutation π and reencrypted using some random values r_1^*,r_2^*, and r_3^*, i.e., $u^*_{\pi(1)} = \mathsf{Enc}(s_{\pi(1)}, r_{\pi(1)} + r^*_{\pi(1)})$, $u^*_{\pi(2)} = \mathsf{Enc}(s_{\pi(2)}, r_{\pi(2)} + r^*_{\pi(2)})$, and $u^*_{\pi(3)} = \mathsf{Enc}(s_{\pi(3)}, r_{\pi(3)} + r^*_{\pi(3)})$. Then, the product over the input blocks $u_1 \cdot u_2 \cdot u_3 = \mathsf{Enc}(s_1 + s_2 + s_3, r_1 + r_2 + r_3)$ encrypts the same value than the product over the output blocks $u_1^* \cdot u_2^* \cdot u_3^* = \mathsf{Enc}(s_1 + s_2 + s_3, r_1 + r_1^* + r_2 + r_2^* + r_3 + r_3^*)$. It follows that the product over the output block is a reencryption of the product over the input block with reencryption value $R_l^{In} = r_1^* + r_2^* + r_3^*$. The association between the input and output blocks is published to provide universal verifiability.

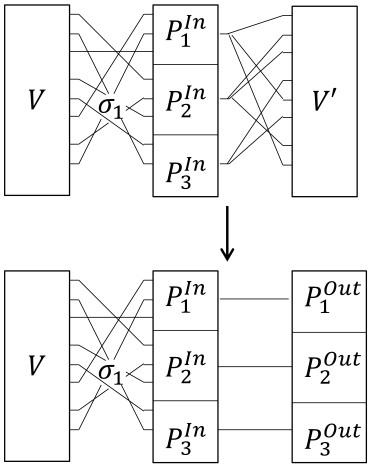


Figure 4.2: Verification procedure for 8 votes after Step 3

4. M_1 has to prove for each block that it has knowledge of the reencryption value R_l^{In} using a zero-knowledge proof of correct reencryption. If, for instance, the encryption scheme ElGamal is used, possible instantiations are the Chaum-Pedersen protocol [CP92] and Schorr's signature scheme [Sch91].

5. After each block has been verified, the output of the second shuffling step V_1 needs to be checked. Furthermore, the blocks must be assigned such that the reencryption of each input of the mix can be contained in each output block. Thus, in the second verification step the division of the input ciphertexts always depends on the blocks built in the first verification step. More precisely, the set of input values V' is divided into L blocks in a way that each input

block $P_l^{In'}$ contains at least one ciphertext of each of the L output blocks P_l^{Out} (see Figure 4.3).

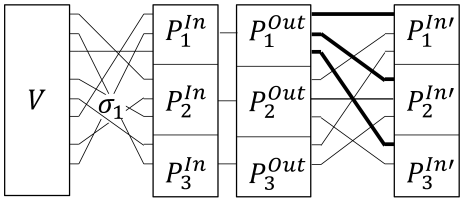


Figure 4.3: Verification procedure for 8 votes after Step 5

6. Mix M_1 proves for each output block $P_l^{Out'}$ that there is an input block $P_l^{In'}$ such that the product of the ciphertexts contained in each block encrypt the same set of votes and that it has knowledge of the corresponding reencryption value $R_l^{In'}$ (see Figure 4.4).

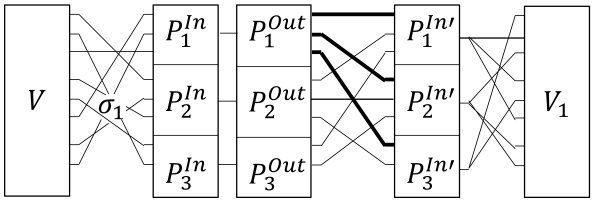


Figure 4.4: Verification procedure for 8 votes after Step 6

7. Step 1 to 6 are repeated for all mixes.

4.4 Analysis and Comparison

In this section we analyze Random Block Verification with respect to correctness, privacy, and efficiency and compare our approach with Randomized Partial Checking, Proof of Subproduct, and the Norwegian Mix-Net.

4.4.1 Correctness

The Random Block Verification procedure is not perfect since an error (e.g., changing a 1 to a 3) can be counterbalanced (e.g., $1 + 4 = 3 + 2$) and pass without being detected. However, to achieve an undetected corruption, a malicious mix has to

change (drop, alter, insert) at least two ciphertexts to ensure that the introduced error is balanced. This will remain undetected only if both adapted ciphertexts end up in the same block. Since the assignment to blocks is not known during shuffling, a malicious mix cannot ensure this. Below, we investigate the probability that 2 modified ciphertexts end up in the same block by chance.

Correctness of our approach can be estimated similarly to the Norwegian Mix-Net since the only difference is the blocksize used which is $\sqrt{K}$ instead of $\sqrt[n]{K}$ for n mixes and K ciphertexts. It follows that the probability of a malicious mix to modify 2 votes without being detected is around $\frac{\sqrt{K}-1}{K-1}$. Note that in reality, it is slightly better as some blocks are smaller than others.

Thus, Random Block Verification provides a better correctness than Randomized Partial Checking and Proof of Subproduct. With respect to the Norwegian Mix-Net the size of the blocks in our verification process does not depend on the number of mixes n and is larger for $n > 1$ (which is usually the case). Thus, the Norwegian Mix-Net is slightly better with respect to correctness since for smaller blocks it is less likely that two ciphertexts end up in the same block.

4.4.2 Privacy

In Random Block Verification, each mix shuffles twice and during verification the K ciphertexts are assigned to $\sqrt{K}$ blocks. So, after the first verification step, the size of the anonymity group (AG) is around $\sqrt{K}$. However, the blocks for the second verification step are built such that they include at least[6] one ciphertext of each of the output blocks of the first verification step. Therefore, to trace the ciphertext through the second verification step, all input blocks need to be considered and the anonymity group has size K.

Compared to common generic verification procedures, like Randomized Partial Checking, Proof of Subproduct, and the Norwegian Mix-Net, Random Block Verification is the only verification process that prevents vote clustering. Furthermore, since in Random Block Verification the assignment for different mixes is independent from each other, this solution is not prone to the attacks identified by Khazaei et al.

[6] If $\sqrt{K} \notin \mathbb{N}$, exactly one ciphertext per block is not possible. However, our approach remains as close as possible to that ideal.

4.4.3 Efficiency

In this section the different approaches are compared regarding efficiency. The computational costs of the verification procedure highly depend on the encryption scheme and zero-knowledge proof of correct reencryption used. In general, the more elements to be checked, the higher the number of operations needed. Thus, the computational costs per mix are estimated by the number of zero-knowledge proofs of correct reencryption that have to be carried out. It follows that Proof of Subproduct is the most efficient scheme since here the correct reencyption of maximal five blocks have to be shown.

Using Randomized Partial Checking half the links of each shuffling step are checked leading to K zero-knowledge proofs. Since we do not show the correct reencryption of each element but of blocks, our approach is more efficient. More precisely, the number of blocks checked using Random Block Verification depends on the size of the input set and is $2\frac{K}{\lfloor\sqrt{K}\rfloor}$ for K inputs and both verification steps. Note that many implementations reveal the reencryption value used instead of proving their knowledge. This is a possible speed up for our approach as well. Nevertheless, the impact of this approach on the security of the verification procedure has not been elaborated so far which is why we do not consider this here. In the Norwegian Mix-Net the efficiency depends on the number of mixes and input ciphertexts. For n mixes and K inputs $\frac{K}{\sqrt[n]{K}}$ blocks are generated and corresponding $\frac{K}{\sqrt[n]{K}}$ reencryption proofs are needed. This makes our scheme also more efficient than the Norwegian Mix-Net (if more than one mix is used).

4.4.4 Summary

Random Block Verification and cut-and-choose are the only generic verification procedures that prevent vote clustering. Although our approach enforces that each mix has to shuffle, reencrypt, and show correct processing of its input set two times, Random Block Verification is still significantly more efficient than cut-and-choose and can also be used for large scale elections. For 1000 votes 31 blocks are generated and thus only 62 proofs of correct reencryption have to be carried out during the verification process. This makes our approach even more efficient than Randomized Partial Checking and the Norwegian Mix-Net.

Furthermore, our approach provides strong assurance that the input batch was processed correctly, i.e., all votes are contained in the output batch. Having, for instance, 1000 votes, a malicious mix will be detected with a probability of 0.9697. Table 4.2 provides and overview of the comparison of Random Block Verification

Table 4.2: Comparison for K ciphertexts and a total number of n mixes regarding correctness (1 ciphertext for RPC and 2 for the other, optimistic mixing based, approaches), privacy, and efficiency per mix node

	RPC	**PoS**	**NM**	**RBV**
Correctness ($P(undetected)$)	$\frac{1}{2}$	$\frac{5}{8}^{\alpha}$	$\frac{\sqrt[n]{K}-1}{K-1}$	$\frac{\sqrt{K}-1}{K-1}$
Privacy ($\|AG\|$ for one honest mix)	$\sim \frac{K}{2}$	$\frac{K}{2^{\alpha}}$	$\sqrt[n]{K}$	K
Efficiency (# ZK-proofs)	K	5	$2\frac{K}{\sqrt[n]{K}}$	$2\frac{K}{\lfloor\sqrt{K}\rfloor}$

(RBV) with Randomized Partial Checking (RPC), Proof of Subproduct (PoS), and the Norwegian Mix-Net (NM) with respect to correctness, privacy, and efficiency.

5 Universally Verifiable Mix-Net Providing Everlasting Privacy

In all known proposals for universally verifiable mix-nets, the audit information is encrypted using some public key encryption algorithm, like ElGamal or Paillier, that are assumed to be computationally hard. However, when the underlying cryptographic assumption is broken for the key length chosen, perhaps decades later, all the audit information can be decrypted and privacy is violated: each output message published can be traced back to its input. With current trends in technology, like quantum computers, such a scenario is realistic. Furthermore, the costs for storing information are decreasing making it virtually impossible to remove (encrypted) information that has been published on the internet. In addition, processing power increases continually following Moore's law. So all an attacker has to do is to wait until the cryptographic assumption for the parameters used is broken, download the audit information from some internet archive, and decrypt it. In other words, the privacy offered by current implementations of mix-nets has an (often unknown) expiration date.

To address this problem, in this chapter we show how the standard shuffling process, described in Section 4.1, can be upgraded to provide everlasting (or unconditional) privacy. The underlying idea is to replace the homomorphic encryptions in the public audit trail with unconditionally hiding and homomorphic commitments. These commitments are rerandomized and permuted by each mix in a publicly verifiable manner. In parallel, their opening values are encrypted with a compatible homomorphic algorithm, reencrypted, permuted, and sent between the mixes over a private channel. Since an unconditionally hiding commitment can be a commitment to any value of the message space with the same probability, such auxiliary information is needed to open the commitments. Thus, for robustness it must be ensured that the permutations and rerandomization values used in the public and in the private channel are the same. The content of this chapter is published in [BDG13].

5.1 Mixing with Everlasting Privacy Towards the Public

5.1.1 Cryptographic Primitives

Our mix-net uses the following cryptographic schemes and zero-knowledge proofs.

Homomorphic Commitment with Matching Encryption Scheme

The protocol uses two cryptographic primitives. First, a commitment scheme $(\mathsf{GenCom}, \mathsf{Com}, \mathsf{Unv})$, as defined in Section 2.2.2, in order to provide everlasting privacy and universal verifiability towards the public. Second, a matching encryption scheme $(\mathsf{GenEnc}, \mathsf{Enc}, \mathsf{Dec})$, as defined in Section 2.2.1, that allows opening the commitment at the end of the reencryption mixing process and at the same time provides privacy towards the authorities.

Note that, because both the vote $s \in \mathcal{M}$ and the decommitment value $t \in \mathcal{R}$ have to be sent over the private channel, we need two instances of the encryption scheme. One, denoted as $(\mathsf{GenEnc}_{\mathcal{M}}, \mathsf{Enc}_{\mathcal{M}}, \mathsf{Dec}_{\mathcal{M}})$, which must be homomorphic over the message space $\mathcal{M}$. The second one, denoted as $(\mathsf{GenEnc}_{\mathcal{R}}, \mathsf{Enc}_{\mathcal{R}}, \mathsf{Dec}_{\mathcal{R}})$, whose message space is homomorphic over group $\mathcal{R}$ of the commitment scheme.

It follows that, having two commitments $u = \mathsf{Com}(s, t)$ and $u' = \mathsf{Com}(s', t')$ and the corresponding opening values in encrypted form $v = \mathsf{Enc}_{\mathcal{M}}(s)$, $w = \mathsf{Enc}_{\mathcal{R}}(t)$, $v' = \mathsf{Enc}_{\mathcal{M}}(s')$, and $w' = \mathsf{Enc}_{\mathcal{R}}(t')$, then the encrypted opening values of two multiplied commitments

$$u \cdot u' = \mathsf{Com}(s + s', t + t')$$

can be computed by multiplying the encryptions, i.e.,

$$\begin{aligned} v \cdot v' &= \mathsf{Enc}_{\mathcal{M}}(s) \cdot \mathsf{Enc}_{\mathcal{M}}(s') = \mathsf{Enc}_{\mathcal{M}}(s + s') \\ w \cdot w' &= \mathsf{Enc}_{\mathcal{R}}(t) \cdot \mathsf{Enc}_{\mathcal{R}}(t') = \mathsf{Enc}_{\mathcal{R}}(t + t') \end{aligned}$$

This allows us to rerandomize a commitment

$$\mathsf{ReRand}(u, t') = u \cdot \mathsf{Com}(0_{\mathcal{M}}, t') = \mathsf{Com}(s, t) \cdot \mathsf{Com}(0_{\mathcal{M}}, t') = \mathsf{Com}(s, t + t')$$

and adapt the encrypted opening values correspondingly

$$\begin{aligned} \mathsf{ReEnc}(v) &= v \cdot \mathsf{Enc}_{\mathcal{M}}(0_{\mathcal{M}}, r'') = \mathsf{Enc}_{\mathcal{M}}(s, r) \cdot \mathsf{Enc}_{\mathcal{M}}(0_{\mathcal{M}}, r'') = \mathsf{Enc}_{\mathcal{M}}(s, r + r'') \\ \mathsf{ReAdj}(w, t') &= w \cdot \mathsf{Enc}_{\mathcal{R}}(t', r''') = \mathsf{Enc}_{\mathcal{R}}(t, r') \cdot \mathsf{Enc}_{\mathcal{R}}(t', r''') = \mathsf{Enc}_{\mathcal{R}}(t + t', r' + r'''). \end{aligned}$$

There are some instantiations that satisfy these properties. Moran and Naor, for instance, show in [MN10, Appendix A] that Paillier encryptions can be used in combination with slightly modified Pedersen commitments.

However, because of Paillier this instantiation is impractical for large elections. More interesting is therefore the recent suggestion of Cuvelier et al. [CPP13] for an efficient unconditionally hiding commitment scheme with matching homomorphic encryption scheme based on elliptic curves. Used in combination with the non-interactive proof (or rather, argument) techniques, proposed by Groth [Gro10] or Lipmaa and Zhang [LZ12] this leads to a very efficient instantiation.

Proof of Correct Shuffling and Consistency

In addition to the proofs of correct shuffling (see Section 2.2.3) this mix-net uses **proofs of consistency**: each mix has to privately prove that the encrypted information and the commitments contained in the output are consistent. More precisely, it has to show that the same permutation and random values have been used to rerandomize the published commitments and to reencrypt the corresponding encrypted opening values. Note that in order to provide everlasting privacy, the published proof showing correct shuffling of the commitments has to be perfect or statistical zero-knowledge. More precisely, it must be guaranteed that if the proof of correct shuffling published by the prover is correct, even a computationally unbounded auditor cannot learn more than that. The requirements for the proof of consistency can be weakened to computational zero-knowledge because the verification process is carried out in private and used to show consistency between commitments and encryptions.

The concrete proofs of correct shuffling and consistency depend on the used instantiations of the encryption and commitment schemes. As mentioned before, for some instantiations correct shuffling can be shown using the perfect zero-knowledge arguments proposed by Groth or Lipmaa and Zhang. Furthermore, also the generic proof presented in Chapter 4 is applicable. Care must be take that the zero-knowledge proof of correct reencryption used to show correct processing of each block is perfect zero-knowledge as well. This can be accomplished by applying, for instance, the "Zero-Knowledge Proof That Two Commitments Are Equivalent" presented in [MN10, Appendix B.1]. This proof can also be used to show consistency between the commitments and encryptions. If $u = \mathsf{Com}(s, t)$ is a commitment to vote s with opening value t and $v = \mathsf{Enc}_{\mathcal{M}}(s, r)$ and $w = \mathsf{Enc}_{\mathcal{R}}(t, r')$ are the corresponding opening values in encrypted form, then consistency can be shown by the following verification process:

1. The prover privately chooses random values $s' \in \mathcal{M}$, $t' \in \mathcal{R}$, and $r'', r''' \in \mathcal{R}'$. Then he or she generates a second set

$$u' = \mathsf{Com}(s + s', t + t')$$
$$v' = \mathsf{Enc}_{\mathcal{M}}(s + s', r + r'')$$
$$w' = \mathsf{Enc}_{\mathcal{R}}(t + t', r' + r''')$$

   and sends the triple $\langle u', v', w' \rangle$ to the auditor.

2. The auditor challenges "0" or "1".

3. If "0", then the prover reveals s', t', r'', and r'''. Now the auditor checks whether:

$$u' \stackrel{?}{=} u \cdot \mathsf{Com}(s', t')$$
$$v' \stackrel{?}{=} v \cdot \mathsf{Enc}_{\mathcal{M}}(s', r'')$$
$$w' \stackrel{?}{=} w \cdot \mathsf{Enc}_{\mathcal{R}}(t', r''')$$

4. If "1", then the prover reveals $s + s', t + t'$, $r + r''$, and $r' + r'''$ and the auditor verifies whether:

$$u' \stackrel{?}{=} \mathsf{Com}(s + s', t + t')$$
$$v' \stackrel{?}{=} \mathsf{Enc}_{\mathcal{M}}(s + s', r + r'')$$
$$w' \stackrel{?}{=} \mathsf{Enc}_{\mathcal{R}}(t + t', r' + r''')$$

If two different values for s or t were used in u, v, and w, then this will be detected with probability $\frac{1}{2}$ unless the prover can solve the computational problem of the cryptographic primitives used. This process is repeated several times until the probability of not being detected is below a certain threshold value. To reduce the communication between the prover and the auditor, a non-interactive version of this process can be obtained by applying the Fiat-Shamir technique [FS86].

5.1.2 Additional Assumptions and Roles

The roles are the same listed in Section 4.1.1: Users, Mixes, Key Trustees, Authorities, Auditors, and Observers. Like the standard verifiable reencryption mix-net, this protocol makes use of a bulletin board to publish the audit information and a trusted Random Beacon to generate the random challenge bits.

Apart from Assumptions (M.1) to (M.4) presented in Section 4.1.2, we make the following additional assumptions:

Assumption M.5 The authorities cannot break the computationally binding property of the commitment scheme for the parameters chosen before the whole mixing process is completed, that is, the votes have been published and certified.

Assumption M.6 There exists a private channel between the voters and the first mix M_1 of the mix-net, between each mix and its successor in the mix-net, and between the last mix and the key trustees.

Assumption M.7 After the protocol has been certified all authorities destroy all information private to them.

5.1.3 Improved Reencryption Mixing Process

In this section we describe how the standard reencryption mixing process can be upgraded to provide everlasting privacy.

Phase I: Vote submission To submit a vote $s \in \mathcal{M}$ encoded with randomness t, during the vote casting process a triple

$$(u, v, w) = (\mathsf{Com}(s, t), \mathsf{Enc}_{\mathcal{M}}(s), \mathsf{Enc}_{\mathcal{R}}(t))$$

is generated. In addition, for each vote, it must be ensured that the s and t used in all three components are the same by a proof of consistency (see Section 5.1.1). The triple (u, v, w) is sent over a private channel to M_1.

The input batch of the first mix consists of all triples received ordered in some canonical way (e.g., lexicographical order) and is denoted as (U_0, V_0, W_0). The public part of the input batch, U_0, is published on the bulletin board.

Phase II: The mixing process We now describe the shuffling procedure for K inputs and a mix-net consisting of n mixes $M_1, M_2, \ldots, M_n$. The input batch of M_j is defined as the output batch of the preceding mix M_{j-1}, except that the input to the first mix are the votes submitted by the voters. In addition, the output batch of the last mix, M_n, will be sent to the key trustees. In other respects, the shuffling procedure for each mix is identical.

1. Let the input batch of mix M_j be $(U_{j-1}, V_{j-1}, W_{j-1})$. Then M_j rerandomizes the commitment vector

$$U'_j = \mathsf{ReRand}(U_{j-1}, T_j) = U_{j-1} \cdot \mathsf{Com}(0, T_j),$$

it reencrypts the vector of encryptions of the votes

$$V'_j = \mathsf{ReEnc}(V_{j-1}) = V_{j-1} \cdot \mathsf{Enc}(0, R),$$

and, through homomorphic encryption, updates the decommit value

$$W'_j = \mathsf{ReAdj}(W_{j-1}, T_j) = W_{j-1} \cdot \mathsf{Enc}_{\mathcal{R}}(T_j, R').$$

2. To obtain the output batch, M_j chooses a random permutation π_j and sets
$$(U_j, V_j, W_j) = \mathsf{Perm}_{\pi_j}(U'_j, V'_j, W'_j).$$

3. The commitments U_j are sent to the bulletin board, whereas the corresponding encryptions V_j and W_j are sent to M_{j+1} through a private channel.

4. M_j proves, in a publicly verifiable way, that U_j is a recoding and permutation of U_{j-1}. This proof of correct shuffling can be verified by the auditors and any observer.

5. M_j provides a proof of consistency and correct shuffling to M_{j+1} showing that the output batch is a consistent rerandomization and permutation of the entire input batch, i.e., it shows that it knows some permutation π_j and some vector of random values R, $R' \in \mathcal{R}$ and $T_j \in \mathcal{M}$ such that

$$(U_j, V_j, W_j) = \mathsf{Perm}_{\pi_j}(U'_j, V'_j, W'_j)$$

$$= \mathsf{Perm}_{\pi_j}(U_{j-1} \cdot \mathsf{Com}(0, T_j), V_{j-1} \cdot \mathsf{Enc}(0, R), W_{j-1} \cdot \mathsf{Enc}_{\mathcal{R}}(T_j, R')).$$

Note that the key trustees verify the output of the last mix M_n.

Phase III: Decoding and publication of the votes The key trustees compute and publish $S^* = \mathsf{Dec}(V_n)$ and $T^* = \mathsf{Dec}(W_n)$.

Observe that due to the homomorphic properties and to the fact that the same permutations π_j have been used in the public and private network, we have that S^* and T^* are the values to open U_n, that is, $U_n = \mathsf{Com}(S^*, T^*)$.

Phase IV: Certification of the output The auditors verify whether $U_n = \mathsf{Com}(S^*, T^*)$. If this condition holds, and all public proofs of correct shuffling hold, then they certify the output. Table 5.1 summarizes the improved mixing process.

Table 5.1: Summary of the mixing process providing everlasting privacy towards the public with commitment scheme C and encryption algorithm $\mathsf{E}_{\mathcal{M}}$ and $\mathsf{E}_{\mathcal{R}}$.

	input from voters	Mix j, for $j \in [1, n]$	output	decoding
public	$U_0 = \mathsf{C}(S_0, T_0)$	$U_j = \mathsf{Perm}_{\pi_j}(U_{j-1} \cdot \mathsf{C}(0, T_j))$	U_n	$U_n \stackrel{?}{=} \mathsf{C}(S^*, T^*)$
private	$V_0 = \mathsf{E}_{\mathcal{M}}(S_0)$	$V_j = \mathsf{Perm}_{\pi_j}(V_{j-1} \cdot \mathsf{E}_{\mathcal{M}}(0))$	V_n	$S^* = \mathsf{D}_{\mathcal{M}}(V_n)$
private	$W_0 = \mathsf{E}_{\mathcal{R}}(T_0)$	$W_j = \mathsf{Perm}_{\pi_j}(W_{j-1} \cdot \mathsf{E}_{\mathcal{R}}(T_j))$	W_n	$T^* = \mathsf{D}_{\mathcal{R}}(W_n)$.

5.2 Properties and Proofs

5.2.1 Correctness, Individual and Universal Verifiability

For proving **correctness**, we need to show that the mixes did not change any of the votes. In other words, we need to show that $S_0 \equiv S_n$, where $\equiv$ stands for the existence of a permutation that maps S_0 to S_n. In our case, correctness does not follow straightaway from existing proofs of mixing schemes since these are all based on encryption, implying that the vote s is unambiguously defined. whereas our scheme uses unconditionally hiding commitments to encode the votes. Since we want correctness to be universally verifiable while preserving everlasting privacy, only the public information can be used. The auxiliary information sent over the private channels only serves to help the key trustees to decode, but are not open to the public.

We will show that, as long as the key trustees can open the commitments that are published by the last mix, i.e., the values S^* and T^* such that $U_n = \mathsf{Com}(S^*, T^*)$, then $S_0 \equiv S_n$. In other words, we show that for correctness it is completely irrelevant *how* the key trustees obtained S^* and T^*. This is a consequence of the fact that the commitment scheme used is computationally binding. Our prove is by way of contradiction, showing that if after running the protocol the output $S_n \not\equiv S_0$, then there exists an efficient algorithm violating the binding property by computing a commitment u that can be opened to two distinct values: $u = \mathsf{Com}(\tau_1, \sigma_1) = \mathsf{Com}(\tau_2, \sigma_2)$.

Theorem 1. *Let $S_0 = \langle s_0(1) \dots s_0(K) \rangle$ be the batch of votes as submitted by the voters, and let $S_n = \langle s_n(1) \dots s_n(K) \rangle$ be the set of votes published by the last mix M_n. Then, under Assumption M.5, and with overwhelming probability, $S_0 \equiv S_n$, even if all authorities conspire to act maliciously.*

Proof: Let us first consider the case that there is only one mix, so $n = 1$. Suppose that M_1 has a strategy A allowing it to create an output batch U_1 such that $S_0 \not\equiv S_1$.

We show how A can be used to create an algorithm B that on input τ_0 returns $u, \sigma_0, \tau_1, \sigma_1$ such that $u = \mathsf{Com}(\tau_0, \sigma_0) = \mathsf{Com}(\tau_1, \sigma_1)$.

B, on input τ_0, creates an input batch by committing to τ_0 using K different values $t_0(i)$, that is, $U_0 = \langle \mathsf{Com}(\tau_0, t_0(i)) \rangle$. Recall that Com is unconditionally hiding, so a poly-time restricted A cannot distinguish this specially constructed input from any arbitrary input. So by assumption, S_1 now contains at least one τ_1 such that $\tau_1 \neq \tau_0$. Let i_1 be the index of τ_1 in S_1. Then we have $u_{i_1} = \mathsf{Com}(\tau_1, \sigma_1)$. Since we know that M_1 on executing its strategy A passed a proof of knowledge successfully, M_1 must know (must be able to extract from A), except with negligible probability, a permutation π_1 between U_0 and U_1, which it can make available to B. Define $i_0 = \pi_1^{-1}(i_1)$, the preimage of i_1 under π_1. Again because of the proof of knowledge, B knows a random number $t_1(i_0)$ such that $u_1(\pi_1(i_0)) = u_0(i_0) \cdot \mathsf{Com}(0, t_1(i_0)) = \mathsf{Com}(\tau_0 + 0, t_0(i_0) + t_1(i_0))$. But at the same time, $u_1(\pi_1(i_0)) = u_1(i_1) = \mathsf{Com}(\tau_1, \sigma_1)$. So by setting $\sigma_0 = t_0(i_0) + t_1(i_0)$, poly-time B has found a commitment u that can be opened in two different ways, contradicting Assumption (M.5).

This argument for one mix can be generalized to a mix of arbitrary size n and an arbitrary coalition of cheating mixes. B can perfectly simulate the behavior of the honest mixes, implying that for each mix M_j, honest or corrupted, the permutation π_j and the random values S_j can be computed. Let i_n be the index with $u_{i_n} = \mathsf{Com}(\tau_1, \sigma_1)$ such that $\tau_1 \neq \tau_0$. Trace τ_1 back through the mix-net by defining recursively $i_{j-1} = \pi_j^{-1}(j)$. So i_j is the index of τ_1 in mix M_j. This means that $\sigma_0 = t^*(i_n) = \sum_{j=0}^{n} t_j(i_j)$ can be computed, and we have found $\tau_0, \sigma_0, \tau_1, \sigma_1$ such that $u_{i_n} = \mathsf{Com}(\tau_0, \sigma_0) = \mathsf{Com}(\tau_1, \sigma_1)$, again a contradiction.□

Note that, because of the audit information published during all the mixing steps, any observer can perform the checks. This, together with the fact that the randomness of the challenge bit is guaranteed by the auditors, means that the protocol is **universally verifiable**. Furthermore, **individual verifiability** follows, simply because of the fact that all voters can check that their input u_0 appears on the bulletin board.

5.2.2 Privacy

Theorem 2. *Under Assumption M.2, the public view, consisting of all the input and output batches of the respective mixes together with the proofs and the information published by the key trustees, reveals no information that allows to link a receipt to a vote cast.*

Proof: If at least one mix is honest and keeps the used permutation secret, then

privacy follows from the fact that the output batch S_n is an unknown permutation of the input batch, S_0. Even a computationally unbounded attacker cannot obtain any additional information, since the commitments used to encode the votes are unconditionally hiding and all used proofs provide perfect zero-knowledge.

5.2.3 Robustness

Theorem 3. *The output of the mix-net can be decoded as long as the consistency of the ballots and the private output set of each mix has been verified.*

Proof: It is clear from the construction that if everybody is honest, then the process must always succeed. Now let us first assume that all the ballots are constructed correctly. Each mix processes its input set and proves correctness of the output to the successor. More precisely, each mix shows that it has knowledge of the permutation and the values used for reencrypting and rerandomizing the input set. Furthermore, they prove that the same recoding values have been used to rerandomize the commitments on the public channel and the encryptions on the private channel. If cheating is detected, the malicious mix or the whole mix-net can simply be replaced.

To avoid that a mix can lie about its private input towards the verifiers, we ask each mix to sign its output. Thus, a verifier can check the authenticity of the input batch. Note that the digital signature scheme used does not need to be information-theoretically secure. In addition to the signature, after the ballot preparation process, the consistency of the public data, i.e., the ballots, and the private data, i.e., the encrypted opening values, must be verified.

5.3 Everlasting Privacy Towards the Authorities

The purpose of the protocol presented in the last section is to introduce a reencryption mix-net that provides everlasting privacy towards the public. However, a weak point in this protocol is the following: the mixes process the triple (u, v, w), where v is an encryption of the vote s. Although private channels are used to communicate the encrypted opening values, this protects the voters from outsiders, but not from the first mix, who will be able to recover s once the computational assumption is broken.

If this is unacceptable, a possible solution is to have several mix-nets in parallel (instead of just one) and have the voters split their votes in various parts and submit each part to a different mix-net. As long as none of the first mixes nor any of their

verifiers share their information with one mix of each mix-net, the privacy of the submitted vote is guaranteed unconditionally.

A problem that arises here is how to match shares that come out of the various mix-nets, needed to recombine the complete vote. We solve this by supposing that a voter submits each share labeled with the same, randomly chosen identification number r, which should be chosen large enough to avoid collision. Since these rs cannot be public (to prevent any of the first mixes or its verifier from tracing back a vote), the vote s and the random identity r are encoded in separate commitments. These are published on the bulletin board and the decommitment values are sent, together with the opening values of the corresponding votes, towards the private mix-nets. After all data have been anonymized by the mix-nets, a new authority $\mathcal{Z}$ matches the shares with the help of the key trustees and the rs.

The same identification number should appear in each of the output batches of the various mixes and $\mathcal{Z}$ should therefore be able to match shares coming from the same voter, reconstruct the vote, and publish it. In addition, $\mathcal{Z}$ can use the published commitments to convince the auditors and observers that the tuples were matched correctly by using a proof of knowledge of the corresponding value r.

5.3.1 Mixing Process Providing Everlasting Privacy Towards the Authorities

We describe the protocol for K inputs and m mix-nets consisting of n mixes with mix-net $l \in [1, m]$ denoted as $M_{l,1}, M_{l,2}, \ldots, M_{l,n}$.

Phase I: Vote submission An encoded vote s consists of the tuple

$$(u, v, w, x, y) = (\mathsf{Com}(s, t), \mathsf{Enc}_{\mathcal{M}}(s), \mathsf{Enc}_{\mathcal{R}}(t), \mathsf{Com}(r, t'), \mathsf{Enc}_{\mathcal{R}}(t')),$$

where u is the commitment to the vote, v and w the corresponding opening values in encrypted form, x a commitment to the identification number r, and y its encrypted decommmitment value t'.

1. The voter generates a random number $r \in \mathcal{M}$.
2. During the vote casting process, the voter splits vote s and the decommitment values t and t' in m shares such that:

$$t = t_1 + t_2 + \cdots + t_m$$

$$t' = t'_1 + t'_2 + \cdots + t'_m$$

$$s = s_1 + s_2 + \cdots + s_m$$

Note that prior to the vote casting process, a subset of ballots must be checked to ensure that r and the shares of s, t, and t' are consistent in the commitments and encryptions.

3. The input to mix l

$$(u_l, v_l, w_l, x_l, y_l) = (\mathsf{Com}(s_l, t_l), \mathsf{Enc}_{\mathcal{M}}(s_l), \mathsf{Enc}_{\mathcal{R}}(t_l), \mathsf{Com}(r, t'_l), \mathsf{Enc}_{\mathcal{R}}(t'_l))$$

is sent to the first mix $M_{l,1}$ using a private channel.

The input batch of the first mix $M_{l,1}$ of mix-net l is denoted as $(U_l, V_l, W_l, X_l, Y_l)$ and the public part U_l and X_l is published together with some receipt identification number on the bulletin board. The voters can and should verify that all shares of their vote appears unmodified, exercising their right to individual verifiability.

Phase II: Mixing Each mix-net l mixes its input batch, consisting of K inputs, similar to the process described in Section 5.1.3.

1. The input batch of mix j of mix-net l is $(U_{l,j-1}, V_{l,j-1}, W_{l,j-1}, X_{l,j-1}, Y_{l,j-1})$. Mix $M_{l,j}$ generates $(U'_{l,j}, V'_{l,j}, W'_{l,j}, X'_{l,j}, Y'_{l,j})$ by computing:

$$\begin{aligned} U'_{l,j} &= \mathsf{ReRand}(U_{l,j-1}, T_{l,j}) \\ V'_{l,j} &= \mathsf{ReEnc}(V_{l,j-1}) \\ W'_{l,j} &= \mathsf{ReAdj}(W_{l,j-1}, T_{l,j}) \\ X'_{l,j} &= \mathsf{ReRand}(X_{l,j-1}, T'_{l,j}) \\ Y'_{l,j} &= \mathsf{ReAdj}(Y_{l,j-1}, T'_{l,j}) \end{aligned}$$

2. To obtain the output batch, $M_{l,j}$ chooses a random permutation $\pi_{l,j}$ and shuffles the data

$$(U_{l,j}, V_{l,j}, W_{l,j}, X_{l,j}, Y_{l,j}) = \mathsf{Perm}_{\pi_{l,j}}(U'_{l,j}, V'_{l,j}, W'_{l,j}, X'_{l,j}, Y'_{l,j}).$$

3. The pair of commitments $(U_{l,j}, X_{l,j})$ is published on the bulletin board, whereas the corresponding private encryptions $V_{l,j}$, $W_{l,j}$, and $Y_{l,j}$ are sent to the next mix $M_{l,j+1}$ through a private channel.
4. $M_{l,j-1}$ uses a proof of correct shuffling to prove towards the public that both sets, $U_{l,j}$ and $X_{l,j}$, are a correct rerandomization and permutation of the corresponding input batch $U_{l,j-1}$ and $X_{l,j-1}$, respectively.

5. $M_{l,j-1}$ privately proves knowledge of the permutation $\pi_{l,j}$ and the random values used to reencrypt the commitments and encryptions such that

$$(U_{l,j}, V_{l,j}, W_{l,j}, X_{l,j}, Y_{l,j}) = \mathsf{Perm}_{\pi_{l,j}}(U'_{l,j}, V'_{l,j}, W'_{l,j}, X'_{l,j}, Y'_{l,j})$$

$$= \mathsf{Perm}_{\pi_{l,j}}(\mathsf{ReRand}(U_{l,j-1}, T_{l,j}), \mathsf{ReEnc}(V_{l,j-1}), \mathsf{ReAdj}(W_{l,j-1}, T_{l,j}),$$
$$\mathsf{ReRand}(X_{l,j-1}, T'_{l,j}), \mathsf{ReAdj}(Y_{l,j-1}, T'_{l,j})).$$

Phase III: Decoding and publication of the votes The tuple $(U_{l,n}, V_{l,n}, W_{l,n}, X_{l,n}, Y_{l,n})$ is considered the final output of mix-net l.

1. With the help of the key trustees, $\mathcal{Z}$ decrypts:

$$S^*_l = \mathsf{Dec}(V_{l,n})$$
$$T^*_l = \mathsf{Dec}(W_{l,n})$$
$$T'^*_l = \mathsf{Dec}(Y_{l,n})$$

   Afterwards, $\mathcal{Z}$ computes $R^*_l = X_{l,n} \cdot \mathsf{Com}(0, -T'^*_l)$ for $l \in [1, m]$ which should decrypt to K equal sets of random numbers R^*, showing the correspondence between tuples coming from the same voter.

2. $\mathcal{Z}$ publishes this correspondence on the bulletin board in the form of a permutations σ_l from $(U_{1,n}, X_{1,n})$ to $(U_{l,n}, X_{l,n})$.

3. $\mathcal{Z}$ privately computes

$$S^* = S^*_1 + \sigma_2(S^*_2) + \cdots + \sigma_l(S^*_l)$$

$$T^* = T^*_1 + \sigma_2(T^*_2) + \cdots + \sigma_l(T^*_l)$$

   which are the values to open

$$\begin{aligned} U^* &= U_{1,n} \cdot \sigma_2(U_{2,n}) \cdot \ldots \cdot \sigma_l(U_{l,n}) \\ &= Com(S^*_1, T^*_1) \cdot Com(\sigma_2(S^*_2), \sigma_2(T^*_2)) \cdot \ldots \cdot Com(\sigma_l(S^*_l), \sigma_l(T^*_l)) \\ &= Com(S^*_1 + \sigma_2(S^*_2) + \ldots + \sigma_l(S^*_l), T^*_1 + \sigma_2(T^*_2) + \ldots + \sigma_l(T^*_l)) \end{aligned}$$

4. For $l \in [2, m]$, $\mathcal{Z}$ also publishes a proof that the commitments

$$X_{1,n}, \ldots, \sigma_l(X_{l,n})$$

   commit to the same identification number R^*_1, i.e.,

$$X_{1,n} = \mathsf{Com}(R^*_1, T'^*_1), \ldots, \sigma_l(X_{l,n}) = \mathsf{Com}(R^*_1, \sigma_l(T'^*_l)).$$

This can be done by proving knowledge of "rerandomization value" $\sigma_l(T_l'^*) - T_1'^*$ and permutation σ_l, because

$$\begin{aligned} X_{1,n} &= \mathsf{Com}(R_1^*, T_1'^*) \cdot \mathsf{Com}(0, \sigma_l(T_l'^*) - T_1'^*) \\ &= \mathsf{Com}(R_1^* + 0, T_1'^* + \sigma_l(T_l'^*) - T_1'^*) \\ &= \mathsf{Com}(R_1^*, \sigma_l(T_l'^*)) \\ &= \sigma_l(X_{l,n}) \end{aligned}$$

Phase IV: Certification of the output The auditors verify whether

$$U^* \stackrel{?}{=} \mathsf{Com}(S^*, T^*).$$

If this condition holds, and all public proofs of correct shuffling hold, then they certify the output. Table 5.2 shows the mixing process for two mix-nets containing of three mixes.

5.3.2 Properties

Correctness, **Individual Verifiability**, and **Universal Verifiability** can be shown similar to the proof in Section 5.2.1. The only difference is that $\mathcal{Z}$ has to prove that votes coming from the same voter have been matched correctly. As mentioned, this is accomplished by a proof of knowledge which shows that when $\mathcal{Z}$ matches, shares coming from different mixes have the same r (see Phase III Section 5.3.1).

With respect to **privacy**, we claim that under the additional assumption that the publication authority $\mathcal{Z}$ does not share its information with any of the first mixes or its verifiers, nor does one of the first mixes or its verifier collaborate with one mix of each mix-net, privacy is also unconditional towards the authorities. But if they do share information then, as long as a sufficient number of key trustees are honest, they still need to break the encryption algorithm Enc.

For **robustness**, if all parties are honest then the protocol terminates successfully, unless for two ballots the same r is chosen, which is an extremely unlikely event. Furthermore, for poll-site voting schemes this can be prevented because the rs are generated by the authorities during the ballot preparation process. Observe that when secret sharing is used, inconsistent IDs can be submitted, for instance, $u_1 = Com(r, t_1')$ to the first mix of the first mix-net and $u_i = Com(r', t_i')$ to the remaining $i-1$ mix-nets, where $r \neq r'$. However, note that in poll-site voting systems the ballot papers are generated by the authorities and a set of ballots is publicly audited. Thus, if inconsistent ballots were generated, this would most likely be detected during the auditing procedure.

Table 5.2: Mixing process for two mix-nets M_j, where $j \in [1,2]$, containing of three mixes A_j, B_j, and C_j with commitment scheme C, encryption algorithm $\mathsf{E}_{\mathcal{M}}$ and $\mathsf{E}_{\mathcal{R}}$, key trustees $\mathcal{K}$, matching authority $\mathcal{Z}$, and bulletin board BB.

private channel	public channel	public channel	private channel
Voter privately splits $s = s_1 + s_2$, $t = t_1 + t_2$, and $t' = t'_1 + t'_2$			
$v_1 = \mathsf{E}_{\mathcal{M}}(s_1)$ $w_1 = \mathsf{E}_{\mathcal{R}}(t_1)$ $y_1 = \mathsf{E}_{\mathcal{R}}(t'_1)$	$u_1 = \mathsf{C}(s_1, t_1)$ $x_1 = \mathsf{C}(r, t'_1)$	$u_2 = \mathsf{C}(s_2, t_2)$ $x_2 = \mathsf{C}(r, t'_2)$	$v_2 = \mathsf{E}_{\mathcal{M}}(s_2)$ $w_2 = \mathsf{E}_{\mathcal{R}}(t_2)$ $y_2 = \mathsf{E}_{\mathcal{R}}(t'_2)$
send to A1	send to BB	send to BB	send to A2
A1's input batch		A2's input batch	
v_1, w_1, y_1	u_1, x_1	u_2, x_2	v_2, w_2, y_2
$v_{A1} = v_1\mathsf{E}_{\mathcal{M}}(0)$ $w_{A1} = w_1\mathsf{E}_{\mathcal{R}}(t_{A1})$ $y_{A1} = y_1\mathsf{E}_{\mathcal{R}}(t'_{A1})$	$u_{A1} = u_1\mathsf{C}(0, t_{A1})$ $x_{A1} = x_1\mathsf{C}(0, t'_{A1})$	$u_{A2} = u_2\mathsf{C}(0, t_{A2})$ $x_{A2} = x_2\mathsf{C}(0, t'_{A2})$	$v_{A2} = v_2\mathsf{E}_{\mathcal{M}}(0)$ $w_{A2} = w_2\mathsf{E}_{\mathcal{R}}(t_{A2})$ $y_{A2} = y_2\mathsf{E}_{\mathcal{R}}(t'_{A2})$
send to B1	send to BB	send to BB	send to B2
B1's input batch		B2's input batch	
v_{A1}, w_{A1}, y_{A1}	u_{A1}, x_{A1}	u_{A2}, x_{A2}	v_{A2}, w_{A2}, y_{A2}
$v_{B1} = v_{A1}\mathsf{E}_{\mathcal{M}}(0)$ $w_{B1} = w_{A1}\mathsf{E}_{\mathcal{R}}(t_{B1})$ $y_{B1} = y_{A1}\mathsf{E}_{\mathcal{R}}(t'_{B1})$	$u_{B1} = u_{A1}\mathsf{C}(0, t_{B1})$ $x_{B1} = x_{A1}\mathsf{C}(0, t'_{B1})$	$u_{B2} = u_{A2}\mathsf{C}(0, t_{B2})$ $x_{B2} = x_{A2}\mathsf{C}(0, t'_{B2})$	$v_{B2} = v_{A2}\mathsf{E}_{\mathcal{M}}(0)$ $w_{B2} = w_{A2}\mathsf{E}_{\mathcal{R}}(t_{B2})$ $y_{B2} = y_{A2}\mathsf{E}_{\mathcal{R}}(t'_{B2})$
send to C1	send to BB	send to BB	send to C2
C1's input batch		C2's input batch	
v_{B1}, w_{B1}, y_{B1}	u_{B1}, x_{B1}	u_{B2}, x_{B2}	v_{B2}, w_{B1}, y_{B2}
$v_{C1} = v_{B1}\mathsf{E}_{\mathcal{M}}(0)$ $w_{C1} = w_{B1}\mathsf{E}_{\mathcal{R}}(t_{C1})$ $y_{C1} = y_{B1}\mathsf{E}_{\mathcal{R}}(t'_{C1})$	$u_{C1} = u_{B1}\mathsf{C}(0, t_{C1})$ $x_{C1} = x_{B1}\mathsf{C}(0, t'_{C1})$	$u_{C2} = u_{B2}\mathsf{C}(0, t_{C2})$ $x_{C2} = x_{B2}\mathsf{C}(0, t'_{C2})$	$v_{C2} = v_{B2}\mathsf{E}_{\mathcal{R}}(0)$ $w_{C2} = w_{B2}\mathsf{E}_{\mathcal{R}}(t_{C2})$ $y_{C2} = y_{B2}\mathsf{E}_{\mathcal{R}}(t'_{C2})$
send to $\mathcal{Z}$	send to BB	send to BB	send to $\mathcal{Z}$
$\mathcal{Z}$'s input batch			
v_{C1}, w_{C1}, y_{C1}	u_{C1}, x_{C1}	u_{C2}, x_{C2}	v_{C2}, w_{C2}, y_{C2}
$\mathcal{K}$ and $\mathcal{Z}$ privately decrypt $y_{C1} \to t'^{*}_1 = t'_1 + t'_{A1} + t'_{B1} + t'_{C1}$		$\mathcal{K}$ and $\mathcal{Z}$ privately decrypt $y_{C2} \to t'^{*}_2 = t'_2 + t'_{A2} + t'_{B2} + t'_{C2}$	
$\mathcal{Z}$ privately decodes $x_{C1} \to \mathsf{C}(r, 0)$		$\mathcal{Z}$ privately decodes $x_{C2} \to \mathsf{C}(r, 0)$	
$\mathcal{Z}$ privately matches the random identities $\mathsf{C}(r, 0)$			
$\mathcal{Z}$ publishes the correspondence as permutation σ			
$\mathcal{Z}$ proves correct matching by showing knowledge of $t'^{*}_2 - t'^{*}_1$			
$\mathcal{Z}$ privately computes $u^* = u_{C1}\sigma(u_{C2})$			
$\mathcal{Z}$ privately computes $v^* = v_{C1}\sigma(v_{C2})$			
$\mathcal{Z}$ privately computes $w^* = w_{C1}\sigma(w_{C2})$			
$\mathcal{K}$ and $\mathcal{Z}$ privately decrypt $v^* \to s^*$			
$\mathcal{K}$ and $\mathcal{Z}$ privately decrypt $w^* \to t^* = t_1 + t_{A1} + t_{B1} + t_{C1} + t_2 + t_{A2} + t_{B2} + t_{C2}$			
$\mathcal{Z}$ publishes $u_{C1}, \sigma(u_{C2}), s^*, t^*$			
$\mathcal{Z}$ and the auditors verify that $u^* = u_{C1}\sigma(u_{C2}) = \mathsf{C}(s^*, t^*)$			

6 Mixing Based Voting Schemes Providing Everlasting Privacy

In this chapter we show how a mixing based voting system can be upgraded to provide everlasting privacy. We do so by describing how the mix-nets presented in Section 5 can be applied to the voting systems Prêt à Voter and Split-Ballot. The results of this section were partially published in [DHG+13].

We start in Section 6.1 with Prêt à Voter because this is one of the best explored electronic voting systems which provides verifiability on the one hand but a familiar and easy way of paper voting one the other hand. Furthermore, this system is planned to be used in the Victoria State Election in Australia in 2014 [BCH+12a, BCH+12b].

With the mix-net described in Section 5.1 Prêt à Voter can be improved such that it provides everlasting privacy towards the public. However, to ensure this towards the authorities as well, additional organizational measures are needed. Thus, in Section 6.2 we discuss how Split-Ballot can be used together with the mix-net described in Section 5.3. Since the ballot paper layout of this scheme allows for secret sharing, we can build a voting system that offers everlasting privacy towards both the public and the authorities.

6.1 Prêt à Voter Providing Everlasting Privacy

Since its introduction in 2004 [RB04], Prêt à Voter has been continuously developed and improved. In [RBH+09], for instance, Ryan et al. describe the key elements and compare two approaches that use different cryptographic primitives. In [RP10] a threat analysis is carried out and enhancements are proposed and in [XCH+10] Xia et al. show how various election methods can be handled. Furthermore, the authors of [DHR+11] analyze the feasibility of Prêt à Voter for German Federal Elections from a technical and legal point of view. However, despite the numerous publications

about Prêt à Voter, this is the first work addressing the aspect of everlasting privacy.

When using a paper based poll-site voting system the mix-net cannot be simply replaced. In this case it is necessary to elaborate how the ballots are generated and printed and how the votes cast can be decoded and counted. Thus, in this section a technical solution for Prêt à Voter is described and evaluated regarding the properties verifiability, everlasting privacy, and robustness.

6.1.1 System Overview of the Classic Scheme

Roles

In addition to the roles listed in Section 4.1.1, the following parties participate in the election process.

Head of the election committee The head of the election committee is responsible for the correct execution of the election procedures. He or she, for instance, opens and closes the vote casting process, overviews the counting of votes, and announces the election result of the constituency.

Poll workers The poll workers (also called *election committee*) assist the head of the election committee during the election procedures, for instance, by checking the eligibility of the voters, handing out ballot papers in the polling station, and supervising the vote casting and auditing process.

Help Organization Help organizations support voters in performing the verification processes.

Clerks The set of clerks is a subset of authorities that generate the ballot data in distributed fashion.

Authorities The set of authorities consists of all private authorities: the head of the election committee, poll workers, key trustees, auditors, and clerks.

System Overview

This section provides a high-level overview of the system and the voter's view. For more information please consult [BCH+12b] or [RP10].

Ballot Form Layout The Prêt à Voter ballot form consists of two halves which can be separated by a perforation down the middle. The lefthand side shows the candidates in random order. The righthand side contains a box against each name

where the voters can mark their choice and includes a link to the used encrypted candidate order (see Fig. 6.1). This can, for instance, be a hash or a serial number which refers to corresponding data published. The information on the righthand side, which allows reconstruction of the candidate order, is hidden by a scratch field.

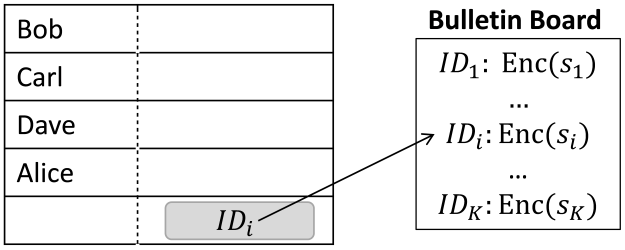


Figure 6.1: Ballot Paper Layout used in Prêt à Voter.

Auditing of Ballot Forms Before the vote casting process, the poll workers publicly audit the well-formedness of a set of randomly selected ballots. This is performed by revealing the scratch fields and verifying that the printed candidate list matches with the encrypted candidate order. In addition, voters should be able to perform their own checks. Thus, each voter receives two or more ballots from which he or she chooses one for vote casting while the remaining ballots are opened and audited.

Vote Casting and Vote Capture The voter authenticates him- or herself, receives a set of ballot papers, and audits all ballots except one. Then, the voter enters the polling booth and votes for a subset of candidates, for instance, by marking the corresponding boxes with an "x". To cast the vote, only the righthand side containing the selected positions and the link to the encrypted candidate order have to be scanned. Thus, before the voter leaves the secret polling booth, he or she detaches and destroys the left hand side, showing the candidate list. Before the ballot is scanned, the poll worker checks whether the scratch field is still intact. If the unique code is revealed during the vote casting process the voter can take a picture of the filled out ballot and prove his or her cast vote towards a coercer. If the scratch field is still intact the poll worker reveals the information at the bottom of the righthand side and scans the ballot paper. Then the scanner displays the digitalization of the receipt (i.e., the righthand side of the ballot) and asks the voter to confirm whether the vote has been recorded correctly. Thus, the voter can cast a

fresh ballot if the information shown does not reflect his or her vote. If the scanned information is correct, the voter confirms. Then his or her encrypted vote is added to the list of votes cast and the voter receives a receipt containing a record of the scanned information signed by the electronic ballot box. In addition, the filled out righthand side is cast to a conventional ballot box allowing to re-scan in case of a malfunction or breakdown. There are also other specifications of Prêt à Voter where the voter keeps the righthand side as a receipt. However, to provide robustness, we recommend to collect them in a conventional ballot box.

Anonymization and Tallying After the vote casting process, all cast encrypted votes, composed of the marked positions and the encrypted candidate order, are anonymized followed by decrypting and tallying. The anonymization process is usually performed with the help of a mix-net (see Section 2.2.3). The input set consists of all published encrypted votes. Then, each mix of the mix-net successively reencrypts the encryptions, i.e., by changing the random values of the ciphertexts, and shuffles its input batch. The output of the mix-net is a set of anonymized encrypted votes which is then decrypted and tallied. For a detailed description of the process please consult Section 4.1.

Verification After the voters received their receipt, they should check the validity of the signature of the scanner[7], for instance, by using a smart phone application or a device which is made available in the polling station. After the vote casting process, all encrypted votes cast are published and each voter can check whether his or her encrypted vote, printed on the receipt, appears. Furthermore, the whole tallying process can be verified by any interested party. More precisely, during mixing, each mix of the mix-net publishes enough information so that the observers are able to check whether the mixing process was performed correctly, i.e., that the input and the output batch encrypts the same set of votes. There are several approaches to prove correct mixing, for instance, by using a non-interactive zero-knowledge argument (see, for instance, [Gro10] or [LZ12]) or a generic verification method (see, for instance, Section 4). In addition, the voting system publishes enough information to allow any observer to verify that the output of the mix-net was decrypted and tallied correctly.

[7]The voters can verify the signature by themselves. Nevertheless, they should have the opportunity to ask poll workers for help.

6.1.2 Technical Details of Prêt à Voter Providing Everlasting Privacy

This section describes the technical details of the proposed voting scheme. The early Prêt à Voter approaches use reencryption mix-nets and support only cyclic shifts of candidate lists [CRS05, RS06]. However, if the voter selects more than one candidate per ballot, the distance between various marks reveals information about the vote cast. Thus, later developments [Hea07, XSH+07, XSHT08] provide arbitrary permutations requiring one encrypted information per candidate. For legibility, we will describe the improved scheme only for ballots containing a shifted candidate order. However, as elaborated for the original approach [XCH+10, SSC+11] the described process can easily be adjusted so that other tallying methods and ballot papers with arbitrary candidate lists are supported.

Additional Roles

If Prêt à Voter is enhanced such that it provides everlasting privacy, there is one additional party that is involved in the election procedure:

Key Server There exists a private key server, for instance, a hardware security module[8] that provides only limited access to authorities. The access key is shared among several key trustees such that no single authority has access to the device. The key server is used to store some key material and is not needed during the vote casting process. Thus, it can be stored safely, for instance, in the town hall.

Assumptions Regarding the Operational Environment

Apart from the assumptions listed in Section 4.1.2 and Section 5.1.2, we make the following assumptions regarding the operational environment.

Assumption O.1 The electoral roll is accurately maintained and voters can cast their vote in a secret polling booth.

Assumption O.2 A non-trivial subset of authorities acts honestly, meaning that they follow the process correctly and do not reveal information private to them, for instance, access or private key shares (compare to Assumption (M.3)

[8] For more information see, for instance, Payment Card Industry (PCI): *PCI Hardware Security Module (HSM)*;`https://www.pcisecuritystandards.org/documents`.

Section 4.1.2). Furthermore, they should be chosen in a way that the probability of collaboration can be kept low, for instance, by selecting members of different parties.

Assumption O.3 The authorities choose the parameters (e.g., keys) for the used cryptographic primitives in a way that the underlying computational problem cannot be broken before the election result has been announced (compare to Assumption (M.1), see Section 4.1.2, and Assumption (M.5), see Section 5.1.2). The public keys for the encryption and commitment scheme should be published on the bulletin board to allow any interested party to verify that the parameters have been chosen properly.

Assumption O.4 All hardware devices need a certification, must be protected by a chain of custody, and have to be tested before use. Furthermore, all electronic systems are prone to "information leakage" (compare to Section 3.1.1). Therefore, the emanations should be evaluated prior to the election by the technique authority and corresponding preventive security measures should be implemented.

Assumption O.5 All private data is safely stored in a key server that is protected by access control, where the keys are shared among a set of key trustees. Furthermore, the device is sealed and stored in a way that it is secure against any access of unauthorized persons.

Assumption O.6 A member of the election committee has to remove the auditing strip and immediately destroy it before the ballot is handed out to the voter. Furthermore, it has to be ensured that the scratch field is intact after the voter left the polling booth.

Assumption O.7 All random values used during the ballot preparation, tallying and, verification phase are unpredictable and random and are produced by a random beacon (compare to Assumption (M.4), see Section 4.1.2).

Assumption O.8 The IT environment provides private channels (compare to Assumption (M.6) Section 5.1.2), which are modification proof and secure against side channel attacks, between

1. the private key server and the first printer,
2. the private key server and the first mix of the mix-net, and
3. successive mixes in the mix-net.

Assumption O.9 After each printing step, the ballot papers are shuffled before they are loaded to the next printer to ensure that neither poll workers nor printers learn the association between IDs and candidate lists.

Assumption O.10 At least one mix of the mix-net is honest and keeps the permutation, used to shuffle its input set, secret (compare to Assumption (M.2) 4.1.2).

Assumption O.11 After processing encrypted data, all hardware components destroy the information private to them (compare to Assumption (M.7) Section 5.1.2).

Assumption O.12 A threshold subset of key trustees attend the tallying process so that the private key server can be accessed and encrypted data can be decrypted.

All election steps, from the ballot preparation to the auditing of the election outcome, should be carried out in public to allow any interested party to observe that:

- Only persons in authority and no single authority access the key server.
- The printing process is performed correctly (Assumption (O.9)).
- Private channels, like direct cable connection, are used for the communication (Assumption (O.8)), and are not subject to manipulation.
- The auditing procedures are carried out and the revealed decommitment values are derived from an appropriate uniform distribution.
- The encrypted opening values are anonymized by a mix-net before they are decrypted.
- All information private to the hardware components are deleted, for instance, by destroying their memory (Assumption (O.11)).

Technical Details

Key generation To provide everlasting privacy and universal verifiability, we need to encode the published auditing information. For this purpose a homomorphic and unconditionally hiding commitment scheme $(\mathsf{GenCom}, \mathsf{Com}, \mathsf{Unv})$ with matching encryption schemes $(\mathsf{GenEnc}_{\mathcal{M}}, \mathsf{Enc}_{\mathcal{M}}, \mathsf{Dec}_{\mathcal{M}})$ and $(\mathsf{GenEnc}_{\mathcal{R}}, \mathsf{Enc}_{\mathcal{R}}, \mathsf{Dec}_{\mathcal{R}})$, as defined in Section 5.1.1, are used. Note that for Prêt à Voter the commitment scheme Com

and encryption scheme $\mathsf{Enc}_{\mathcal{M}}$ must be additively homomorphic with respect to the message space. Furthermore, the message space must be large enough to prevent an exhaustive search on the possible shift and commitment values published.

Prior to an election, parameters for the commitments and two key pairs are generated by executing GenCom, $\mathsf{GenEnc}_{\mathcal{M}}$, and $\mathsf{GenEnc}_{\mathcal{R}}$. The public keys $\mathsf{pk}_{\mathcal{M}}$ and $\mathsf{pk}_{\mathcal{R}}$ are published while the corresponding private keys $\mathsf{sk}_{\mathcal{M}}$ and $\mathsf{sk}_{\mathcal{R}}$ are distributed in threshold fashion among several key trustees. During the ballot printing process three printers are used. The second printer generates a key pair $(\mathsf{sk}_{\mathcal{R}'}, \mathsf{pk}_{\mathcal{R}'})$ and the third printer a key pair $(\mathsf{sk}_{\mathcal{M}'}, \mathsf{pk}_{\mathcal{M}'})$ using $(\mathsf{GenEnc}_{\mathcal{R}})$ and $(\mathsf{GenEnc}_{\mathcal{M}})$ respectively. The public keys are published together with the other key material.

Ballot Preparation For the Prêt à Voter voting system providing everlasting privacy, the conventional ballot layout is adopted but instead of the encrypted shift value $\mathsf{Enc}(s)$ the ballot refers to a commitment $\mathsf{Com}(s,t)$ of the used candidate order. Furthermore, the ballot is extended by an "auditing" strip, containing the decommitment value t, which can be detached by a perforation (see Figure 6.2). After the

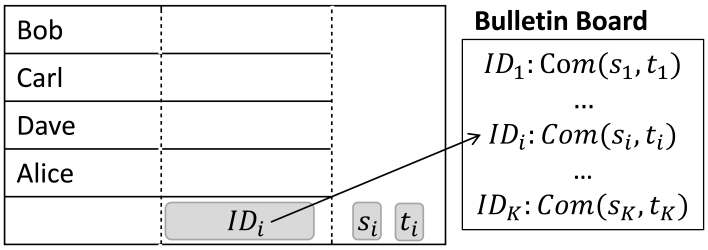


Figure 6.2: Ballot Paper Layout of the new version.

voter audited some ballots and selected one for vote casting, a member of the election committee removes the auditing strip of the remaining ballot and immediately destroy it. If the voter would be able to detach and keep the auditing strip, he or she can reveal the used decommitment value and prove a certain candidate order towards an attacker.

After vote casting, all scanned ballots are published showing the position marked by the voter and a unique commitment to the shifted candidate list. To provide vote secrecy, no single authority or electronic device must be able to reconstruct the association between the published information and the corresponding secret candidate order. Thus, similar to the original approach, the ballot data (i.e., shifted candidate order, commitment, and encrypted opening values) is generated in distributed fashion by several clerks. To generate a set of K ballots, each clerk $j \in [1, L]$ performs the following steps:

1. Generate a batch of random seeds $\left\langle s_i^j \in \mathcal{M} \right\rangle_{i=1}^{K}$ denoting a cyclic shift of the candidate names. The position of the candidate with respect to the initial candidate ordering can be determined by computing modulo m, where m is the number of candidates. The seeds are drawn uniformly at random from the message space.

2. A commitment to each seed i is generated: $\left\langle \mathsf{Com}_{ck}(s_i^j, t_i^j) \right\rangle$, using a randomly chosen decommitment value $t_i^j \in \mathcal{R}$.

3. The opening values are encrypted with the public keys $\mathsf{pk}_{\mathcal{M}}$ and $\mathsf{pk}_{\mathcal{R}}$ of the key trustees, $\langle \mathsf{Enc}_{\mathcal{M}}(s_i^j), \mathsf{Enc}_{\mathcal{R}}(t_i^j) \rangle$, and the public keys $\mathsf{pk}_{\mathcal{R}'}$ and $\mathsf{pk}_{\mathcal{M}'}$ of the second and third printer, $\langle \mathsf{Enc}_{\mathcal{M}'}(s_i^j), \mathsf{Enc}_{\mathcal{R}'}(t_i^j) \rangle$.

Each clerk sends its output to the key server and proves consistency between the commitments and encryptions using a "proof of consistency" (see Section 5.1.1). If the proofs hold, the private key server does the following operations on its input:

1. It generates the "full" encrypted information $(\Theta_i^C, \Theta_i^E, \Theta_i^S, \Theta_i^T)$ for ballot i by multiplying the output of various clerks:

$$\begin{aligned}
\Theta_i^C &= \prod_{j=0}^{L} \mathsf{Com}_{ck}(s_i^j, t_i^j) \\
\Theta_i^E &= \left\langle \prod_{j=0}^{L} \mathsf{Enc}_{\mathcal{M}}(s_i^j), \prod_{j=0}^{L} \mathsf{Enc}_{\mathcal{R}}(t_i^j) \right\rangle \\
\Theta_i^S &= \prod_{j=0}^{L} \mathsf{Enc}_{\mathcal{M}'}(s_i^j) \\
\Theta_i^T &= \prod_{j=0}^{L} \mathsf{Enc}_{\mathcal{R}'}(t_i^j).
\end{aligned}$$

2. The private key server generates a link, ID_i, for each ballot $i \in [1, K]$ and publishes the IDs together with the commitments $\{\Theta_i^C\}_{i=1}^{K}$ and a commitment to the corresponding encrypted opening values on the bulletin board.

3. Then the set of IDs, $\{\mathrm{ID}_i\}_{i=1}^{K}$, is sent together with $\{\Theta_i^S\}_{i=1}^{K}$ and $\{\Theta_i^T\}_{i=1}^{K}$ to the first printer while the corresponding opening values $\{\Theta_i^E\}_{i=1}^{K}$ are kept secret by the key server.

Ballot Printing The generated ballot data is printed by a quorum of printers (see Figure 6.3), similar to the process described in [RP10]. Another possibility is to print the ballots on demand in the polling-station. However, a legal analysis showed that printing should be carried out in advance [DHR+11].

1. The first printer prints the encrypted seed Θ^S on the lefthand side of the ballot paper, the link ID to the commitment Θ^C at the center, and the encrypted decommitment value Θ^T on the righthand side.

2. The printed ID is covered by a scratch field, the ballot papers are shuffled, and loaded into the next printer.

3. The second printer scans and decrypts $\Theta^T = \mathsf{Enc}_{\mathcal{R}'}(t)$ and prints the decommitment value t at the bottom of the righthand side. Then t is covered by a scratch field, Θ^T is removed, the ballot papers are shuffled, and loaded into printer three.

4. The last printer scans and decrypts the seed $\Theta^S = \mathsf{Enc}_{\mathcal{M}'}(s)$ and prints the candidate list shifted by $-s \pmod m$ on the lefthand side. Furthermore, it prints the seed value on the auditing strip next to t. Finally, s is covered by a scratch strip, and the encrypted seed value Θ^S is removed.

Note that to ensure voter privacy, Assumption (O.2), (O.8), (O.9), and (O.11) must hold. Furthermore, the ballots should be printed in public to assure that the described process is performed correctly. If, for instance, the printing process is not observable, a printer could remove all scratch fields, read and store the encrypted information, and use this data to violate voter privacy. Even worse is if this attack is carried out by the third printer, since this machine is able to decrypt the seed values and thus does not have to wait until the cryptosystem is broken.

Even though the organizational measures ensure voter privacy if only one printer is used, the ballots should be printed in distributed fashion. By doing so, the relevant information is not located on one device and a successful attack is harder to carry out. To manipulate a printer successfully, one either needs to have access to the device within the limits of the certification, or has to get possession of the printer in the polling station. This is more difficult in case more than one printer is used. To make successful attacks even harder, additional technical measures can be considered. The replacement of printed ballots, for instance, can be made more difficult by watermarks or fingerprints[9] on the ballots which allow verifying their integrity.

[9]Note that if fingerprints are used which make the ballot papers unique, the corresponding information must be hidden or destroyed before the ballots are handed out to the voters.

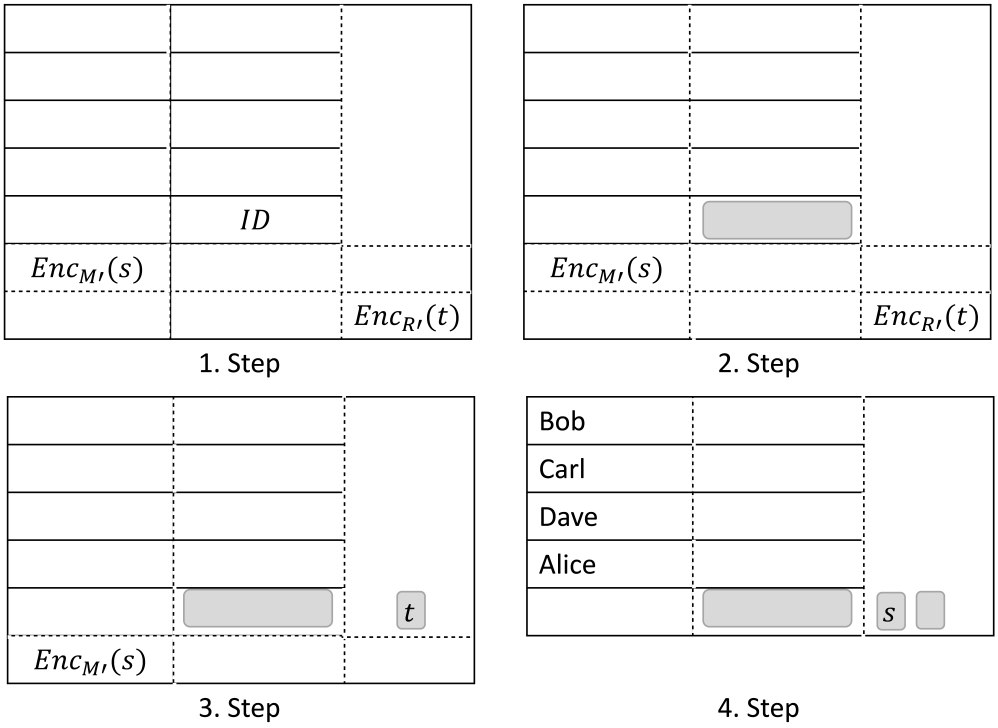


Figure 6.3: Ballot printing process.

Auditing Process Auditing of ballot forms is very important for the robustness of the voting system. If the ballot papers were not generated properly, the votes cast are not decoded correctly afterwards. To check the well-formedness of a ballot paper, one just has to reveal the opening values s and t and the link to the commitment $\mathsf{Com}(s,t)$ hidden under the scratch fields. Then, the verifier can check whether the value s modulo m was used to shift the candidate names and that s and t open the commitment $\mathsf{Com}(s,t)$. This proves integrity of the ballot due to the computational bindingness of the commitment scheme used.

Before the vote casting process, the authorities choose a subset of ballots at random that is publicly audited. Besides the well-formedness, the auditors should check that the revealed shift and decommitment values were derived from an appropriate uniform distribution. Furthermore, the key server has to reveal the encrypted opening values. By decrypting the ciphertexts the authorities can verify that the ballot provides consistency between the commitments and encryptions.

Note that the commitments printed on the ballot forms are published during the tallying process. Thus, if voters get to see the decommitment and shift value, they can use this information to open the commitment, prove the candidate order of their

ballot, and thus whom they cast their vote for. Therefore, if a ballot paper is used to cast a vote, the auditing strip, containing the hidden opening values s and t, **must** be detached and destroyed by the poll workers before the voter enters the secret polling booth. Furthermore, ballot papers used for auditing **must not** be used for vote casting.

Anonymization, Tallying, and Verification Process Like in the conventional Prêt à Voter tallying process the marked position p_i on each cast ballot $i \in [1, K]$ is publicly encoded and homomorphically added to the commitment, that is

$$\mathsf{Com}(s_i, t_i) \cdot \mathsf{Com}(p_i, 0) = \mathsf{Com}(s_i + p_i, t_i) = \mathsf{Com}(s'_i, t_i).$$

Note that the marked position and the shift value add up to the position of the chosen candidate with respect to the initial, unshifted candidate list modulo m. Then the commitment to the vote cast $\mathsf{Com}(s'_i, t_i)$ is published next to the scanned ID and marked position on the bulletin board. In addition, the private key server adapts the securely stored encrypted shift values accordingly:

$$\mathsf{Enc}_{\mathcal{M}}(s_i) \cdot \mathsf{Enc}_{\mathcal{M}}(p_i) = \mathsf{Enc}_{\mathcal{M}}(s_i + p_i) = \mathsf{Enc}_{\mathcal{M}}(s'_i).$$

Afterwards, the votes are anonymized with the help of the mix-net introduced in Section 5.1.1. The public commitments, $U = \mathsf{Com}(S', T)$, and the privately stored opening values consisting of the encrypted votes, $V = \mathsf{Enc}_{\mathcal{M}}(S')$, and the encrypted decommitment values, $W = \mathsf{Enc}_{\mathcal{R}}(T)$, are sent to the first mix using a private channel. The mix-net publicly outputs a set of mixed commitments, $U^* = \mathsf{Com}(S^*, T^*)$, and privately outputs a set of associated anonymized encrypted votes, $V^* = \mathsf{Enc}_{\mathcal{M}}(S^*)$, and encrypted decommitment values, $W^* = \mathsf{Enc}_{\mathcal{R}}(T^*)$. Note that correctness of the mixing process can be universally verified. After the mixing process, the votes can be decoded and published without violating voter privacy, because the link between single inputs and single outputs has been removed. To determine the election outcome, first, the key trustees decrypt and publish the votes $S^\dagger = \mathsf{Dec}_{\mathcal{M}}(V^*)$ and decommitment values $T^\dagger = \mathsf{Dec}_{\mathcal{R}}(W^*)$. Then any interested party can verify that these are the opening values to the published commitments:

$$U^* = \mathsf{Com}(S^*, T^*) \stackrel{?}{=} \mathsf{Com}(S^\dagger, T^\dagger) = \mathsf{Com}(\mathsf{Dec}_{\mathcal{M}}(V^*), \mathsf{Dec}_{\mathcal{R}}(W^*)).$$

Second, the chosen candidates are determined by publicly computing $S^\dagger \pmod m$, revealing the position marked with respect to the initial, unshifted candidate ordering. Finally, the number of votes each candidate received are counted and the election outcome is announced.

6.1.3 Security Properties

Similar to the classic Prêt à Voter voting scheme, the improved voting protocol provides robustness, correctness, verifiability, computational privacy, and receipt-freeness. Furthermore, our version provides everlasting privacy towards the public and, with the help of certain organizational measures, also towards the authorities.

Robustness

Regarding robustness, the only difference between the classic Prêt à Voter voting scheme and the solution proposed here is the use of a key server to store the opening values. Without this information and the presence of a sufficient number of key holders (Assumption O.12) the votes cast cannot be determined. Therefore, a backup and recovery concept must be developed which provides a high security standard. However, in some applications even a small probability that the ongoing election might be disturbed is unacceptable. For such cases a two layer ballot paper, as proposed for Punch Scan [FCS06], can be used. The upper layer shows the secret candidate order and the bottom layer contains the ID that links to the encoded shift value. The voters cast their choice on the top sheet and the marked positions are recorded by the bottom due to holes punched in the upper layer (see Figure 6.4). After filling out the ballot, the top layer reflects the vote in plain text and is

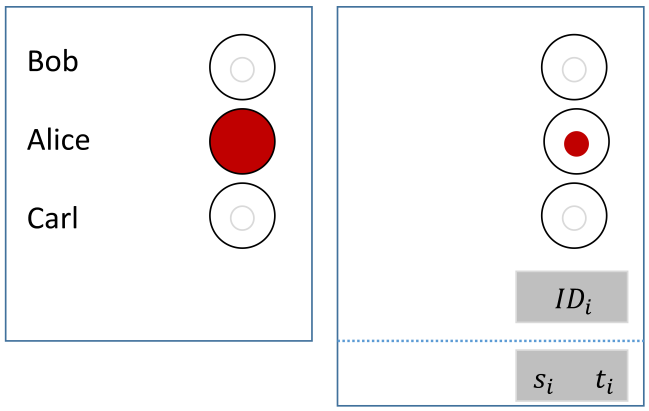


Figure 6.4: Two layer ballot paper with filled out top (l) and bottom layer (r).

collected in a conventional ballot box. The bottom sheet contains the same vote in encoded form and is scanned and publicly processed. Thus, in case of a malfunction or breakdown of the key server, the ballots cast on the conventional ballot box can be used to determine the election result.

Correctness and Verifiability

The proposed voting system provides correctness under the assumption that the computational problem of the used commitment scheme cannot be broken before the outcome has been computed (Assumption O.3) and that the challenges for the performed verification procedures are random and unpredictable (Assumption O.7). Furthermore, the system allows all voters to verify that their vote was encoded as intended, cast as encoded, recorded as cast, and tallied as recorded (see Section 2.1.2 and 2.1.3). If the voter successfully audited some ballots, correctness of the ballot preparation process is ensured with high probability, i.e., the candidate order matches the information the ID refers to. The auditing process also allows them to ascertain themselves that using these ballots the vote is **encoded as intended**. After scanning the encoded vote, the scanner shows its interpretation to the voter and prints a receipt if he or she confirms. Thus, the voter can check that the filled out ballot matches the vote printed on the receipt and therefore that the vote is **cast as encoded**. Furthermore, after the vote casting process, the entire input of the tally is published. Thus, each voter can verify that his or her encoded vote printed on the receipt is **stored as cast**, i.e., that it is included in the input batch of the tallying process.

Furthermore, the used mix-net provides correctness and universal verifiability (see Section 5.2.1) and after the anonymization process, the decoded votes are publicly tallied. Thus, any interested party can recount the election outcome and ascertain itself that the votes were **tallied as recorded**.

Note that we only require integrity to be guaranteed at the moment of vote casting and tallying. However, as discussed with respect to robustness, in the case of Prêt à Voter the ballot layout can be designed in a way that a plaintext vote and a vote in encoded form are generated simultaneously. This allows to collect the filled-out ballots representing the plaintext vote in a conventional ballot box for recount.

Privacy

To analyze the privacy of the proposed system, we will distinguish between receipt-freeness, privacy towards the public, and privacy towards the authorities (see Section 2.1.1). An overview of the addressed vulnerabilities described in Section 3 is given in Table 6.1.[10]

[10]Note that we do not discuss the so called Randomization attack or Italian attack, where voters are forced to mark one or more specific positions on the ballot. This attack has a similar effect than to force a voter to abstain from voting why this violates the principle of free suffrage and not of the principle of secret suffrage. Furthermore, this attack can be confronted by offering

Receipt-freeness The auditing strip is destroyed before the ballot is handed out to the voter and it is checked that the remaining scratch field is intact after the voter left the secret polling booth (Assumption O.6). Thus, he or she cannot reveal the opening values used and prove a certain candidate order to a coercer. Furthermore, the voter cannot disclose the unique code and take a photo of the filled out ballot to prove his or her voting decision cast towards an attacker.

Privacy Towards the Public The proposed voting system provides everlasting privacy towards observers because only the voter gets to see the secret candidate order (Assumption O.1). Furthermore, all data printed on the receipts and published on the bulletin board for auditing, like commitments and zero-knowledge proofs, provide unconditional privacy.

Privacy Towards the Authorities **Computational privacy** towards the authorities is achieved by the encryption schemes used. Their parameters are chosen such that the opening values cannot be decrypted during the election (Assumption O.3) except by the key holders. To minimize the risk of a manipulated authority, the private keys are shared among several persons in authority. Furthermore, to prevent that they conspire to act maliciously, they are chosen in a way that the probability of collaboration can be kept low (Assumption O.2). Furthermore, a single point of failure during the printing or anonymization procedure is prevented (Assumption O.9 and O.10).

Under the organizational assumptions listed in Section 6.1.2, this voting system also provides **everlasting privacy** towards authorities. They prevent that an attacker is able to access and store the ciphertexts until he or she is able to break the encryption scheme used for the key length chosen. More precisely, the encrypted opening values are safely stored in a key server (Assumption O.5) which is protected by access control. Furthermore, only private channels[11] are used for communication what protect the encryptions from eavesdropping (Assumption O.8).

The printers and scanners see the opening values in encrypted form, which only provides computational privacy. Therefore, all hardware devices are tested, certified and protected by a chain of custody (Assumption O.4). This prevents that

voters the chance to ask for a new ballot form. In Germany, according to § 56.8 FEC, the returning committee is permitted to hand out a new ballot paper. Thus, voters would be able to cast their vote for the focused candidate as well as for the positions the intruder wants them to select.

[11] Note that if CDs are used, they must be publicly destroyed immediately after the first printer or mix read out the data.

they contain unauthorized soft- or hardware that allows information to be leaked or data to be manipulated. After they processed the data, all information private to the hardware components is deleted (Assumption O.11), for instance, by physically destroying the memory. This prevents that an intruder can get access to the encryptions after the election. Note that a legal evaluation showed that making these assumptions to provide everlasting privacy is acceptable from a legal point of view. For more information see [DHG+13].

Table 6.1: Overview of addressed privacy vulnerabilities presented in Section 3

Vulnerability	**Addressed by**	**Assumptions**
(Computational) Voter Privacy		
Printer Knowledge	Distributed printing process	system and organizational
Information Leakage	- *(votes are processed in encoded form)*	-
Storing Information	- *(votes are processed in encoded form)*	-
Fingerprints	- *(ballots are used for vote casting)*	-
Printouts	- *(votes are recorded in encoded form)*	-
Single Point of Failure	- *(votes are processed in encoded form)*	-
Authority Knowledge	Threshold decryption	system
	Distribution of access keys	system
	Selecting key trustees from different parties	organizational
Vote Clustering	All cast votes are mixed before decrypting	system
	Usage of privacy preserving shuffling proofs	system
Receipt-Freeness		
Unique Ballots	Scratch field	system
	Poll workers remove auditing strips before handing out a ballot paper for voting	organizational
	Poll workers check scratch fields before scanning a filled out ballot paper	organizational
Everlasting Privacy		
Towards Observers	Publishing unconditionally hiding commitments and perfect zero-knowledge proofs only	system
Towards Authorities	Encrypted data is stored in a certified key server	organizational
	Key server is securely stored and protected by access control	organizational
	Encrypted data is sent over private channels	organizational
	Electronic devices delete all information private to them	organizational
	All used electronic devices are certified	organizational
	Chain of custody for electronic devices	organizational
	All electronic devices are tested prior to using	organizational

6.2 Improving the Split-Ballot Voting Scheme

As Chapter 6.1.3 showed, to provide everlasting privacy towards authorities, certain organizational measures are needed. A comparison with traditional paper based elections and the interpretation of the relevant jurisdiction of Germany showed that such procedural controls are acceptable from a legal point of view (for a detailed legal analysis of this aspect please consult [DHG+13]).

However, there might be other elections whose legal regulations enforce that everlasting privacy towards the authorities must be fulfilled by the voting system itself. For this purpose, we propose a solution that is based on the Split-Ballot voting system. This approach enforces a more complex vote casting procedure but prevents a single point of failure with respect to everlasting privacy.

Although the classic scheme already provides everlasting privacy, the election outcome is jointly computed by two authorities only. Thus, voter privacy is provided under the assumption that at least one of these two authorities act honestly and do not reveal information private to him or her. This is a drawback compared to other verifiable voting systems where the number of authorities sharing secret information is scalable[12].

Thus, in this section we propose an improved voting system that provides not only everlasting privacy but also scalability with respect to computational privacy.

6.2.1 The Classic Split-Ballot Voting Scheme

In this section we start with an introduction to the classic Split-Ballot voting system from the voters' point of view, i.e., the ballot layout and vote casting process. Later, we provide technical details about the ballot preparation, vote recording, and tallying procedure and, finally, summarize its properties and drawbacks with respect to privacy.

The Voters' View

Like Prêt à Voter, the Split-Ballot voting system encodes votes by recording the position marked in a permuted list of candidates. To provide voter privacy, the permutation value used is kept secret. We assume that only one vote is cast per ballot since this scheme only allows shifted candidate lists. How the ballot layout can be generalized to an arbitrary candidate order is out of scope and part of future work.

[12]Note that the vote itself is not shared.

Ballot Layout The ballot paper consists of three separate pages, the top page, the middle page, and the bottom page.

Bottom Page The bottom page is used to record the encoded votes and contain boxes for each candidate where the voters can mark their choice (see Figure 6.5). When all three pages are overlayed, these bubbles are visible through holes punched in the middle and top page.

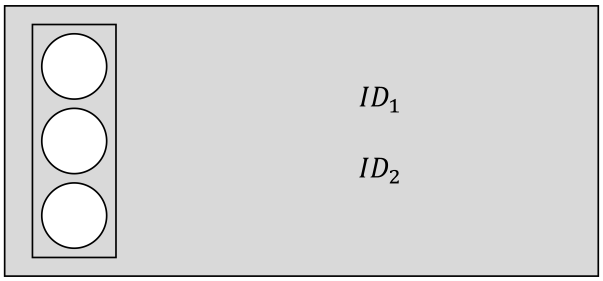


Figure 6.5: Bottom page for three candidates.

Middle Page The middle page contains a hole showing the boxes printed on the bottom page. Furthermore, it shows m distinct lists of candidates, where m is the number of candidates and each list is a permutation of the initial candidate ordering using a random shift value. To optimize the space needed for the lists, shortcuts for each voting option, letters or symbols, are printed instead of the full identifier (see Figure 6.6).

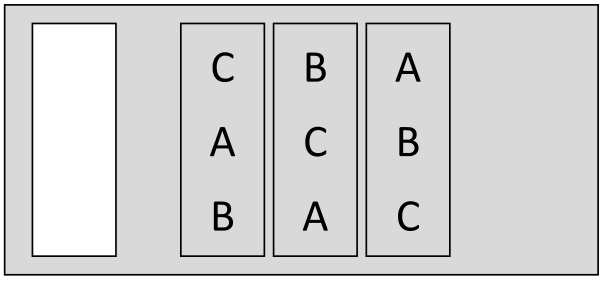


Figure 6.6: Middle page for three candidates.

Top Page The top page contains two holes: one showing the boxes printed on the bottom sheet and one which selects one candidate list printed on the middle page. Furthermore, the top page provides an overview of the shortcuts used on the middle page and the associated voting options (see Figure 6.7).

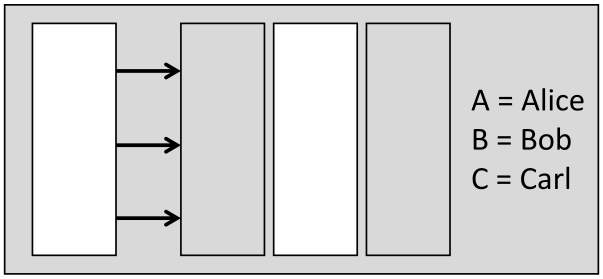


Figure 6.7: Top page for three candidates.

Vote Casting Process Each polling station receives a set of top and middle pages. The bottom pages are generated on demand.

1. The voter enters the polling station and authenticates him- or herself. Afterwards, he or she receives two envelopes, one marked with "top" containing a top page and one marked with "middle" containing a middle page. Each envelope is provided with an ID, printed on a sticker, that links to a commitment published on the bulletin board. More precisely, the "top" envelope links to a commitment that commits to the position of the hole punched in the top page. The commitment the "middle" envelope refers to commits to the shift values used to generate the candidate lists on the middle page.
2. If the voter wants to audit the ballot, then the envelopes are opened, the top and middle page are scanned, and the voter keeps the pages. After the vote casting process, the voting system publishes the opening values to the audited ballots on the bulletin board and the voter can verify the correctness of the ballots[13].
3. If the voter wants to use the ballots to cast a vote, the poll worker removes the stickers from the two unopened envelops and affixes them to a bottom page.
4. The voter enters the secret polling both, opens all envelopes, and overlays the pages. Figure 6.8 shows the voters' view using the top page shown in Figure 6.7 and the middle page shown in Figure 6.6. Afterwards, the voter fills out the ballot by marking the box next to the candidate of his or her choice.

[13]If the commitments are inconsistent with the values used to generate the top and middle page, the voting system would not be able to open them due to the computational binding property of the commitment scheme.

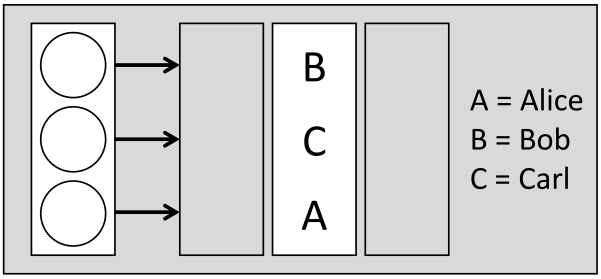


Figure 6.8: Complete ballot paper containing of top, middle, and bottom page.

5. The voter destroys the top and middle page and leaves the secret polling booth. Figure 6.9 shows the filled out bottom layer of the ballot (Figure 6.8) reflecting a vote for Bob.
6. The bottom page, that consists of the IDs referring to the used top and middle page and the position marked by the voter, is scanned. Furthermore, the voter receives a printout of the encoded vote cast as receipt.

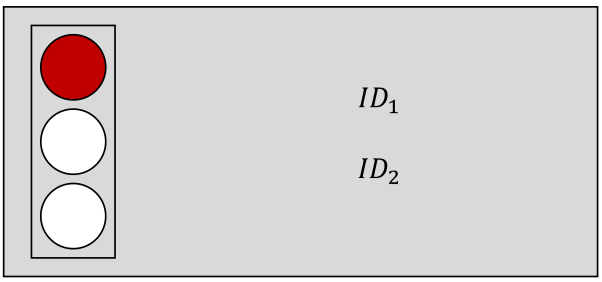


Figure 6.9: Filled out bottom page.

As long as it is unknown which values were used to generate the candidate lists printed on the middle page and which list was chosen by the top page, the position marked by the voter reveals no information about the candidate selected. It follows that the vote cast by the voter is secretely shared among two authorities, authority A_1 that generated the top page and authority A_2 that generated the middle page.

Technical Details

In this section we provide technical details of the ballot preparation, vote recording, and tallying procedure.

Ballot Preparation The voting system Split-Ballot uses the same primitives as the improved Prêt à Voter scheme and the mix-net proposed in Section 5: The public audit trail is encoded with a commitment scheme $\mathsf{Com}(s,t)$, where $s \in \mathcal{M}$ is a vote share and $t \in \mathcal{R}$ a random decommitment value. Furthermore, two matching homomorphic encryption schemes, $\mathsf{Enc}_{\mathcal{M}}$ and $\mathsf{Enc}_{\mathcal{R}}$ are used to privately process the encrypted opening values s and t. Note that the commitment scheme Com and the encryption scheme $\mathsf{Enc}_{\mathcal{M}}$ must be additively homomorphic with respect to the message space.

For each top page,

1. A_1 chooses a shift value $s_1 \in \mathcal{M}$ that denotes which list from the middle page is selected.
2. It commits to this value by calculating $\mathsf{Com}(s_1, t_1)$, using a randomly chosen decommitment value $t_1 \in \mathcal{R}$.

The commitment is considered to be public and will be published next to the marked position on the bulletin board. The used shift value s_1 must be kept secret to ensure voter privacy and will be shown to the voter only. Figure 6.10 provides an overview of all possible top pages generated for three candidates.

The second authority A_2 creates the data for the middle pages. Having m candidates it draws up m distinct candidate lists by

1. choosing a permutation value s_2 at random,
2. generating list $j \in [1, m]$ by shifting the initial list with shift value $s_2 + j$, and
3. calculating a commitment $\mathsf{Com}(s_2 + j, t_j)$ for each list, where t_j denotes a value chosen uniformly at random.

Figure 6.11 shows all possible middle pages for an election with three candidates.

Vote Recording Assume the voter uses the ballot shown in Figure 6.8 and the initial candidate ordering is (Alice # 0, Bob # 1, Carl #2). The second hole on the top page is punched leading to the shift value $s_1 = 1$. Furthermore, the **first list** on the middle page (Carl, Alice, Bob) was generated by shifting the initial candidate ordering using permutation value $s_2 = 1$.

It follows that the shift value needed to reconstruct the initial ordering out of the list shown to the voter can be computed by

$$s \equiv -s_1 - s_2 \pmod{m}.$$

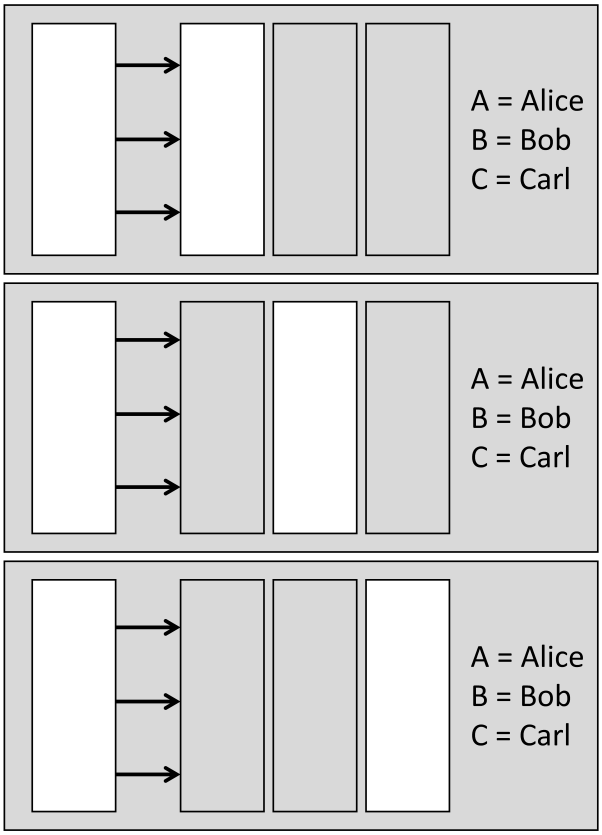


Figure 6.10: Possible top pages for three candidates selecting the first list (top), second list (middle), and third list (bottom) printed on the middle page.

For the shift values used in our example s is

$$1 \equiv -1 - 1 \pmod 3.$$

After the voter filled out the ballot by marking the box at position p, the selected candidate with respect to the initial candidate ordering s' can be calculated by

$$s' \equiv p + s \pmod m.$$

If the voter chooses $p = 0$, as shown in Figure 6.9, s' is

$$1 \equiv 0 + 1 \pmod 3,$$

a vote for Bob with respect to the initial candidate ordering (Alice # 0, Bob # 1, Carl #2).

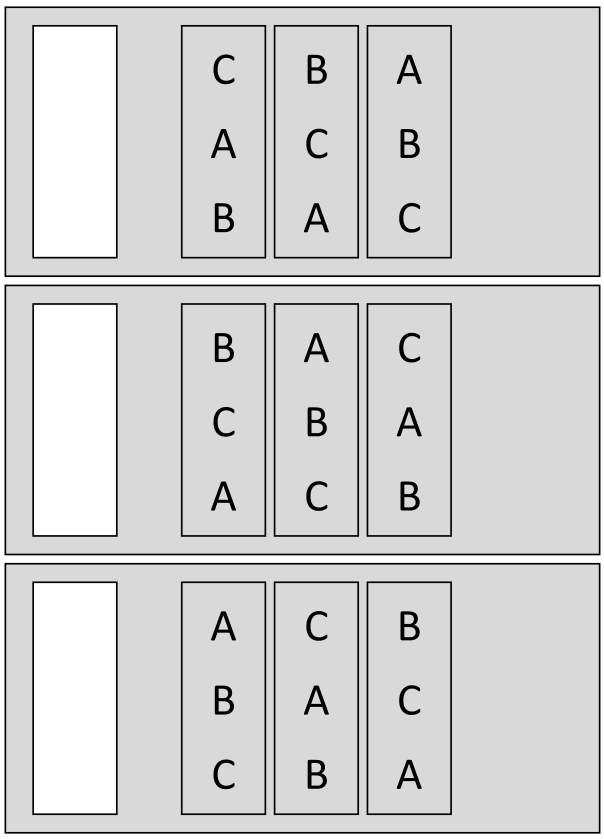


Figure 6.11: Possible middle pages for three candidates using permutation value 1 (mod 3) (top), 2 (mod 3) (middle), and 0 (mod 3) (bottom) to generate the first list.

Vote Tallying After the vote casting process has ended, both authorities, A_1 and A_2, jointly compute the election outcome without disclosing each other the corresponding shift values. This can be accomplished by using the two encryption algorithm, $\mathsf{Enc}_{\mathcal{M}}$ and $\mathsf{Enc}_{\mathcal{R}}$, to privately process the opening values. For a detailed explanation of the tallying procedure please consult [MN10].

Properties

In the classic Split-Ballot voting system the ballot papers are jointly generated and tallied by two authorities. While the **correctness** of the election outcome is independent of the trustworthiness of them, for **voter privacy** certain assumptions are made. As stated by Moran and Naor in [MN10], under the assumption that both authorities are honest and keep their private information secret, the system provides

receipt-freeness[14] and **everlasting privacy**. If only one authority is honest, the system still provides **computational privacy** but may no longer be receipt free. In the case that both authorities are corrupt, privacy is no longer guaranteed.

It follows that, comparing this scheme with other verifiable poll-site voting systems, Split-Ballot has a drawback regarding the ability to violate voter privacy as long as the computational assumptions hold. In Split-Ballot two authorities are enough to reveal the association between the receipt IDs and votes cast. Conversely, in many other verifiable poll-site voting schemes, the subset of malicious authorities needed to break computational privacy is scalable. In Prêt à Voter[15] the number of dishonest authorities required for an attack depends, for instance, on the number of key holders needed to decrypt information, the number of clerks that participate in the ballot generation process, and the size of the mix-net. Using (k, n)-threshold decryption $n - k + 1$ honest authorities out of n key holders are needed to prevent that encrypted data can be disclosed unauthorized. Furthermore, only one trustworthy clerk is enough to prevent that the association between the IDs and the shift values can be revealed during the ballot generation process. The same holds true for mixing. One honest mix out of all mixes is enough to prevent that receipts can be linked to encrypted and counted votes.

It follows that, assuming the participation of a set of l clerks and a mix-net of size m, at least $\min(k, l, m)$ malicious authorities are needed to violate voter privacy as long as the computational assumptions hold. Note that, although everlasting privacy is a very important design goal, it is more harmful to reveal the association between receipts and votes cast during or shortly after the election.

Furthermore, scalability with respect to the number of authorities has several advantages. First, it allows choosing the authorities is a way that the probability of collaboration can be kept low and, second, allows distributing trust among different political parties. In Germany § 9.2 sent 4 Federal Electoral Act regulates, that the election committee shall represent the parties of the respective electoral district (see legal analysis in [DHG+13]). Consequently, the set of authorities should be composed of members of all political parties. This is in conflict with Split-Ballot where privacy relies on the trustworthiness of two authorities only.

[14] Since the IDs are not printed on the middle and top page, they do not have to be hidden by scratch fields. The voters can make a photo of the filled out ballot but can fake this proof by using another set of middle and top pages for vote casting.

[15] We assume that a version is used where the ballots are printed in distributed fashion. Furthermore, we assume that the execution of the election procedures are carried out in public such that authorities and observers can verify that the processes are carried out correctly, i.e., Assumption A, D, F, and I hold.

6.2.2 Improved Split-Ballot Voting Scheme

To improve Split-Ballot, we propose to replace the tallying procedure with the mix-net introduced in Section 5.3.1. With respect to the ballot preparation and vote casting process only minor changes are needed.

Ballot Preparation We use the standard Split-Ballot paper layout. The top and middle page should be independently generated and printed in distributed fashion, as described for Prêt à Voter in Section 6.1.2.

Similar to the standard Split-Ballot voting system, for each middle page, a set of authorities A_2 chooses a shift value s_2 uniformly at random. This value is used to generate the shuffled candidate lists. Independently a second set of authorities A_1 generates the data for the top pages. This is another batch of shift values, where each value s_1 determines at which position the hole is punched in the top layer and consequently which candidate list of the middle page is shown to the voter. Furthermore, each envelope containing a top or middle page is marked with an internal ID that helps to match vote shares after the anonymization process (compare to Section 5.3.1).

Summarized, the public information for each top ($i = 1$) and middle page ($i = 2$), contains of

- commitment $c_i = \mathsf{Com}(-s_i, t_i)$ to the shift value s_i and random decommitment value t_i,
- commitment $\mathsf{Com}(r_i, t'_i)$ to the internal ID r_i and random decommitment value t'_i, and
- IDs, where each ID_i refers to a commitment c_i.

Furthermore, to be able to open the unconditionally hiding commitments, the opening values have to be processed in private. For the top page this data is stored in a key server K_1. Analogously, the information needed for the commitments associated to the middle pages are managed by a second key server K_2. This includes the following data:

- The encrypted opening values $\mathsf{Enc}_{\mathcal{M}}(-s_i)$ and $\mathsf{Enc}_{\mathcal{R}}(t_i)$ for each commitment c_i and
- the decommitment value t'_i for each internal ID r_i in encrypted form $\mathsf{Enc}_{\mathcal{R}}(t'_i)$.

Vote Casting After the voter filled out the ballot, the position p of the box marked and ID_1 and ID_2 are scanned. Afterwards, a commitment to p with decom-

mitment value 0 is publicly generated and homomorphically added to the shift value associated to the top page:

$$\mathsf{Com}(p, 0) \cdot \mathsf{Com}(-s_1, t_1) = \mathsf{Com}(p - s_1, t_1).$$

The key server K_1 adapts its stored information accordingly by calculating

$$\mathsf{Enc}_{\mathcal{M}}(p) \cdot \mathsf{Enc}_{\mathcal{M}}(-s_1) = \mathsf{Enc}_{\mathcal{M}}(p - s_1).$$

Vote Tallying The vote shares, consisting of $\mathsf{Com}(p - s_1, t_1)$ and the shift value used on middle page $\mathsf{Com}(s_2, t_2)$, are processed independently by two different mix-nets M_1 and M_2. To provide that the commitments can be shuffled independently while ensuring that they can be matched afterwards, the association between both inputs must be marked. Furthermore, to ensure receipt-freeness, this ID should be unknown to single authorities. Thus, for each voter the internal IDs of both pages, top page and middle page, are publicly added

$$\mathsf{Com}(r_1, t_1') \cdot \mathsf{Com}(r_2, t_2') = \mathsf{Com}(r_1 + r_2, t') = \mathsf{Com}(r, t')$$

and the resulting ID r marks vote shares coming from the same voter. Afterwards, all encoded votes and their IDs are anonymized by performing the following steps:

1. The public input to mix-net M_1 are two sets of commitments

$$U_1 = \mathsf{Com}(P - S_1, T_1)$$
$$X_1 = \mathsf{Com}(R, T').$$

In addition M_1 privately receives

$$V_1 = \mathsf{Enc}_{\mathcal{M}}(P - S_1)$$
$$W_1 = \mathsf{Enc}_{\mathcal{R}}(T_1)$$
$$Y_1 = \mathsf{Enc}_{\mathcal{R}}(T_1')$$

from K_1 and

$$Y_2 = \mathsf{Enc}_{\mathcal{R}}(T_2')$$

from K_2.

Thus, the private input set of M_1 constitutes of the set of encrypted opening values V_1, W_1, and the decommitment value

$$Y_1' = Y_1 \cdot Y_2 = \mathsf{Enc}_{\mathcal{R}}(T_1') \cdot \mathsf{Enc}_{\mathcal{R}}(T_2') = \mathsf{Enc}_{\mathcal{R}}(T')$$

2. Analogously, the public input to mix-net M_2 is

$$U_2 = \mathsf{Com}(-S_2, T_2)$$
$$X_2 = \mathsf{Com}(R, T')$$

and the private input is

$$V_2 = \mathsf{Enc}_{\mathcal{M}}(-S_2)$$
$$W_2 = \mathsf{Enc}_{\mathcal{R}}(T_2)$$
$$Y_2' = Y_1 \cdot Y_2.$$

3. Both mix-nets shuffle their inputs.
4. The public output of mix M_1 is

$$U_1^* = \mathsf{Com}(-S_1^*, T_1^*)$$
$$X_1^* = \mathsf{Com}(R_1^*, T_1^{*\prime})$$

and the private output is

$$V_1^* = \mathsf{Enc}_{\mathcal{M}}(-S_1^*)$$
$$W_1^* = \mathsf{Enc}_{\mathcal{R}}(T_1^*)$$
$$Y_1^* = \mathsf{Enc}_{\mathcal{R}}(T_1^{*\prime}).$$

5. Analogous, the public output of mix M_2 is

$$U_2^* = \mathsf{Com}(-S_2^*, T_2^*)$$
$$X_2^* = \mathsf{Com}(R_2^*, T_2^{*\prime})$$

and the private output is

$$V_2^* = \mathsf{Enc}_{\mathcal{M}}(-S_2^*)$$
$$W_2^* = \mathsf{Enc}_{\mathcal{R}}(T_2^*)$$
$$Y_2^* = \mathsf{Enc}_{\mathcal{R}}(T_2^{*\prime}).$$

6. Using the encrypted decommitment values Y_1^* and Y_2^*, the output is matched by a special authority $\mathcal{Z}$. It denotes the association by a permutation π which is published on the bulletin board. Furthermore, $\mathcal{Z}$ proves correct matching in perfect zero-knowledge fashion.
7. The opening values are revealed by calculating:

$$S^* = \mathsf{Dec}_{\mathcal{M}}(V_1^* \cdot \pi(V_2^*)) = \mathsf{Dec}_{\mathcal{M}}(\mathsf{Enc}_{\mathcal{M}}(P^* - S_1^* - \pi(S_2^*)))$$
$$T^* = \mathsf{Dec}_{\mathcal{R}}(W_1^* \cdot \pi(W_2^*)) = \mathsf{Dec}_{\mathcal{R}}(\mathsf{Enc}_{\mathcal{R}}(T_1^* + \pi(T_2^*)))$$

8. The set of decoded votes S^* is published on the bulletin board.
9. The set of decrypted decommitment values T^* is published and each voter and observer can verify correctness of the set of vote S^* by verifying whether:
$$U_1^* \cdot \pi(U_2^*) \stackrel{?}{=} \mathsf{Com}(S^*, T^*)$$

6.2.3 Security Properties

Since we only replaced the anonymization procedure with the mixing process presented in Section 5.3.1, our improved Split-Ballot voting system still fulfills the requirements **correctness**, **individual verifiability**, **universal verifiability**, and **receipt-freeness**.

Furthermore, our improvement provides **robustness** under the same assumptions as the standard Split-Ballot voting system. Since a subset of the generated ballots can be audited prior to the election, the consistency between the commitments and the encrypted opening values can be ensured with high probability. In addition, the IDs used to match vote shares are publicly computed, which is why neither malicious authorities nor voters can submit inconsistent IDs undetected. Thus, under the assumption that the encrypted data stored in K_1 and K_2 can be accessed (Assumption O.12), votes cast can be decoded and tallied.

Similar to the original approach our improvement provides **everlasting privacy**. One computational unbounded authority is not enough to reveal the association between receipt IDs and the corresponding votes cast. But unlike Split-Ballot, as long as the encryption scheme holds for the key length chosen, the number of malicious authorities needed to violate voter privacy is scalable. Thus our scheme is an improvement with respect to **computational privacy**. Similar to Prêt à Voter, the size of the subset of dishonest authorities needed depends on the parameters of the threshold cryptosystem, the number of clerks, and the size of the mix-net. This allows to choose the authorities and their responsibilities in a way that the probability of a successful attack is low.

7 Everlasting Private Voting Schemes Using Homomorphic Tallying

There are two widely used approaches how an election outcome can be determined if votes are recorded in encrypted form. The encrypted votes are either anonymized by shuffling and reencrypting followed by decrypting and tallying, or the election outcome is generated by homomorphic tallying and decrypted afterwards.

After we described in Chapter 5 how votes can be anonymized providing universal verifiability and everlasting privacy, in this chapter we focus on the second approach. We, first, show how the primitives used for mixing, a homomorphic commitment scheme with matching homomorphic encryption scheme, can be used to build a homomorphic tallying process that provides universal verifiability and everlasting privacy. Second, we look at Scratch & Vote, the only poll-site voting system that uses homomorphic tallying, and describe how this scheme can be improved. The content of Section 7.1 of this chapter is published in [DGA12].

7.1 Universally Verifiable Homomorphic Tallying Process

In this section we show how votes can be homomorphically tallied providing universal verifiability and everlasting privacy. First, we describe how the votes can be encoded, second how correctness of the encoding can be proven, and third how the votes cast can be tallied. Finally, we provide an overview of the assumptions made and the properties achieved.

As shown in [DGA12], this protocol can easily be applied to online voting systems, like Helios. For legibility, we assume that the voter interacts with a *Direct Recording Electronic Voting Machine* (DRE). How this process can be applied to scan based voting is discussed in Section 7.2.

7.1.1 Encoding of Votes

We represent the vote for candidate i as a vector $\langle s_1, ..., s_m \rangle$ which is 0 everywhere, except in the ith position, where it equals 1. Each entry is encoded individually using the encoding scheme introduced in Section 5.1.1.

More precisely, the commitment $u = \mathsf{Com}(s, t)$ encodes the vote s by "blinding" it with a random number t. Since the used commitment scheme is perfectly hiding, the vote cannot be determined from the commitment itself. Thus, to provide decoding, the opening values s and t are encrypted with a matching homomorphic encryption scheme $\mathsf{Enc}_{\mathcal{M}}(s)$ and $\mathsf{Enc}_{\mathcal{R}}(t)$ and are privately processed.

So the encoding of vote s has three components.

$$\langle u, v, w \rangle = \langle \mathsf{Com}(s, t), \mathsf{Enc}_{\mathcal{M}}(s), \mathsf{Enc}_{\mathcal{R}}(t) \rangle$$

Note that we obtain the following homomorphic properties. If $\langle u', v', w' \rangle = \langle \mathsf{Com}(s', t'), \mathsf{Enc}_{\mathcal{M}}(s'), \mathsf{Enc}_{\mathcal{R}}(t') \rangle$, then

$$\langle u, v, w \rangle \cdot \langle u', v', w' \rangle = \langle \mathsf{Com}(s + s', t + t'), \mathsf{Enc}_{\mathcal{M}}(s + s'), \mathsf{Enc}_{\mathcal{R}}(t + t') \rangle$$

Note also that the commitment Com and the encryption scheme $\mathsf{Enc}_{\mathcal{M}}$ must be **additive homomorphic** with respect to the message space $\mathcal{M}$. This allows us to homomorphically tally the encrypted votes $\prod \mathsf{Enc}_{\mathcal{M}}(s)$ in private and prove correctness of the result by opening the outcome of the publicly processed commitments $\prod \mathsf{Com}(s, t)$.

7.1.2 Proof of correct encoding

To avoid cheating and to provide verifiability and robustness, we need to perform the following public and private verification processes.

Public Verification Process

The observers and auditors must be able to check that the encoded vector indeed represents a valid voting decision and that all votes are tallied correctly. More precisely, it needs to be shown that:

1. Each entry s_i is either 0 or 1. This can be proven using the "Zero-Knowledge Proof that a Committed Value is in $\mathbb{Z}_{2^k}$" proposed by Moran and Naor for their Split-Ballot voting system [MN10, Appendix B.3].

2. If the voter is allowed to choose X candidates per ballot, it must be shown that at most X entries equal 1. More precisely, it must be proven that $\sum_{i=1}^{n} s_i \leq X$. Using homomorphic commitments, this can be achieved by computing the product of all commitments $\prod_{i=1}^{n}(\mathsf{Com}(s_i, t_i)) = \mathsf{Com}(s', t')$ and showing that $s' \leq X$ using the "Zero-Knowledge Proof that a Committed Value is in $\mathbb{Z}_{2^k}$".

3. The votes were tallied correctly. This can be shown by publicly tallying the commitments and opening the commitment to the election result afterwards. This also proves that the opening values, i.e., the number of votes per candidate and the sum or product of the decommitment values, were determined correctly. If this is not the case, then the authorities are not able to open the commitment due to the computational binding property of the commitment scheme used.

Note that all proofs have to be perfect zero-knowledge or witness hiding to ensure voter privacy even in the presence of a computationally unbounded attacker. Furthermore, they must be published to allow any third party to check correctness of the proofs.

Private Verification Process

The private verification process only addresses the robustness of the election scheme since correctness must be publicly verifiable. To be able to open the commitments later on, the same values s and t must be used in the commitment and encryptions. Therefore, the authority who generates these values must privately prove this fact by showing knowledge of vote s, the random decommitment value t, and the randomness used to generate the encryptions. The possible instantiations for this proof depend on the used encryption and commitment scheme. However, a standard cut-and-choose based verification protocol, as described in Section 5.1.1 can be used in any case.

7.1.3 Tallying Protocol

In this section we describe a universally verifiable homomorphic tallying process that provides everlasting privacy towards the public. Let there be m candidates and K voters, and let $i \in [1, m]$ range over all candidates and $j \in [1, K]$ range over all voters. Then for each candidate i the following steps are performed. An overview is given in Figure 7.1

Phase 1: System initialization

1. During the election setup, the system's parameter for the encryption and commitment scheme are generated by executing GenCom and Gen (see Section 2.2.2 and 2.2.1). The secret keys for the encryption scheme used are shared in threshold fashion among several persons in authority, while the corresponding public keys are published.

Phase 2: The voter's perspective

1. Voter j chooses a candidate.

2. The DRE chooses decommitment values $t_1, \ldots, t_m$ and computes the commitments and encryptions

$$\langle \mathsf{Com}(s_1, t_1), \mathsf{Enc}_{\mathcal{M}}(s_1), \mathsf{Enc}_{\mathcal{R}}(t_1), \ldots, \mathsf{Com}(s_m, t_m), \mathsf{Enc}_{\mathcal{M}}(s_m), \mathsf{Enc}_{\mathcal{R}}(t_m) \rangle$$

$$= \langle \overline{u}(j), \overline{v}(j), \overline{w}(j) \rangle.$$

3. The DRE shows the commitment $\overline{u}(j)$ to the voter and he or she can challenge the encoding. In this case the used decommitment values are revealed and the voter can verify the correct encoding either by recalculating or by using software provided by a trusted third party (e.g., a mobile app). An audited ballot cannot be cast, so he or she is asked to fill out a fresh one.

4. If the voter decides to cast the encoded vote, the DRE sends the commitments and encrypted opening values to the electronic ballot box. Note that private channels should be used to send the encryptions since they provide computational privacy only. After the vote cast is stored in the ballot box, the commitment is printed and handed out to the voter as receipt.

5. The DRE must perform the verification processes as explained in 7.1.2. It published the commitments $\overline{u}(j)$ together with a zero-knowledge proof that this is a valid vote on the bulletin board. Furthermore, the device needs to privately prove towards the voting system (e.g., the electronic ballot box) that the encrypted opening values are consistent with the commitments, i.e., s and t are the same in the encryptions and commitments.

6. **(Individual verifiability)** As soon as the polling stations closed, the voters can go to the bulletin board, type in their receipt ID, and verify that the commitments to their vote $\overline{u}(j)$ is included in the list of votes cast.

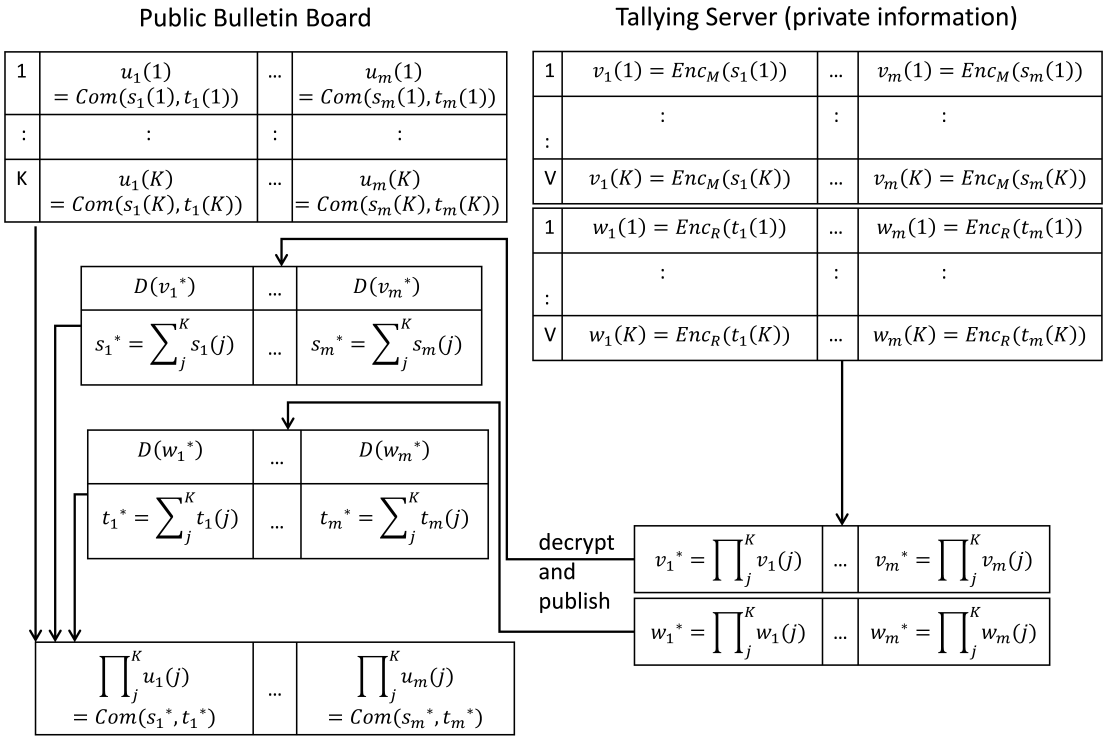


Figure 7.1: Information published on the bulletin board (left) and the privately processed data (right).

Phase 3: Tallying and publishing the votes

1. The system computes $v_i^* = \prod_j v_i(j)$ and $w_i^* = \prod_j w_i(j)$, decrypts v_i^* to $s_i^* \mathrel{\widehat{=}} \sum_j s_i(j)$ and w_i^* to $t_i^* \mathrel{\widehat{=}} \sum_j t_i(j)$ with the help of the key trustees and publishes the result on the bulletin board.

2. The system publicly computes $u_i^* = \prod_j u_i(j) \mathrel{\widehat{=}} \mathsf{Com}(s_i^*, t_i^*)$.

Phase 4: Verification of the tally – universal verifiability

For each vote cast, the ID of the receipt and the value $\overline{u}_j = \langle u_1(j), \ldots, u_m(j) \rangle$ must appear on the bulletin board. After the election, auditors and observers can check the correctness of the tally, as follows:

1. For each i the verifier checks whether the votes were aggregated correctly by computing $u_i^\dagger := \prod_j u_i(j)$ and verifying whether $u_i^\dagger \stackrel{?}{=} u_i^*$.

2. The verifier checks whether $u_i^\dagger \stackrel{?}{=} \mathsf{Com}(s_i^*, t_i^*)$ using the opening values s_i^* and t_i^* published by the system.

7.1.4 Assumptions and Properties

Our scheme relies on the following assumptions.

1. Assumption (O.4) as defined in Section 6.1.2: The authorities running the electronic voting system cannot break the computational problem for the parameters chosen before the elections ends.

2. There exists a private channel that can be used to send encrypted information (compare to Assumption (O.7) Section 6.1.2).

Based on these assumptions our scheme offers the same properties as standard systems using homomorphic tallying, but with everlasting privacy towards the public.

Correctness Even if all authorities conspire, correctness of the election outcome is guaranteed. This is a consequence of the computational binding property of the commitment scheme used.

Individual Verifiability The voters can verify the correctness of the ballot construction because of Phase 2.2 in Section 7.1.3, which allows them to challenge the DRE and to verify the encoding. If each voter verifies one encoding, then a cheating system has probability of $\frac{1}{2}$ of being caught for each wrongly constructed ballot. Furthermore, all voters are able to verify that their vote appears on the bulletin board. As presented in Phase 2.4, $\overline{u}(j)$ is published and thus each voter can verify that his or her encoded vote is contained in the tally (Phase 2.5).

Universal Verifiability Any observer can verify that the tally was computed correctly. This follows immediately from the correctness and from the checks described in Phase 4.1 and 4.2. Once correctness of u_i^* has been verified, and it has been confirmed that t_i^* is the decommitment value of the encoded sum s_i^*, the election result s_i^* must be correct (under the computational assumption of the commitment scheme).

Everlasting Privacy Towards the Public The protocol provides everlasting privacy towards the public, because the commitment scheme used is perfectly hiding. So an encoded vote published on the bulletin board can be any voting decision with equal probability. Furthermore, all proofs presented to the voter to

verify correctness are perfect zero-knowledge, so do not reveal any information about the vote cast. To reveal the voting decision submitted by a voter, an attacker needs additional information besides the data published by the electronic voting system.

Robustness The sum of the aggregated votes can be decoded as long as the proof of consistency provided by the DRE has been verified and a threshold number of authorities attend the decryption process.

Since the opening values, consisting of the vote and the decommitment value, are processed in encrypted form, this system does not provide everlasting privacy towards authorities. Nevertheless, this property can, similar to the mix-net based approach, be provided by secret sharing or organizational measures. We will discuss this in the next section.

7.2 Scan Based Voting System Using Homomorphic Tallying

There is only one paper based voting system that uses homomorphic tallying to determine the election result, Scratch & Vote [AR06]. In this section we will first give an overview of the classic scheme. Afterwards, we show how it can be adapted such that it provides everlasting privacy and, finally, discuss the properties of the improved version.

7.2.1 Voting Scheme Providing Everlasting Privacy Towards the Public

Regarding the ballot layout, Scratch & Vote and Prêt à Voter use the same idea to generate encrypted votes. The voters mark their preference on ballots that contain the voting options in a random and secret ordering. Consequently, both schemes have a similar ballot preparation and vote casting process. The main difference lies in the encoding and tallying of the votes cast[16].

In this section we assume that the candidate lists are generated by shifting the initial candidate ordering. However, similar to Prêt à Voter, this process can be

[16]In addition, in [AR06] the authors propose a better auditing process. The random values used to generate the encryptions are printed on an audit strip that is covered by a scratch field. This allows checking the well-formedness of the ballots without having access to the private key.

generalized to provide arbitrary candidate list[17].

The Scratch & Vote Voting Scheme

Vote Encoding Since the candidate ordering differs from ballot to ballot, the position marked by the voter does not reveal any information about the candidate selected. Thus, we cannot apply the protocol presented in Section 7.1. Instead, the entire ballot must be encrypted in one ciphertext.

The tallying procedure of Scratch & Vote is based on "homomorphic counters" (formalized by [KMO01]) which works as follows: Suppose an election has m candidates and K eligible voters. Then the bitspace of the plaintext is partitioned in m separate counters of size L, $\{2^0, 2^L, \ldots, 2^{(m-1)L}\}$, such that $K \leq 2^L$.

Each candidate $i \in [0, m-1]$ is assigned to a counter 2^{iL} and a bitspace from bit iL to bit $(i+1)L-1$ of the plaintext. Each time a candidate receives a vote, his or her corresponding counter is incremented.

Example. Assume we have three candidates Alice #0, Bob #1, and Carl # 2, and each candidate has a bitspace of $L = 4$. Adding 2 votes for Alice, 3 votes for Bob, and 1 vote for Carl leads to the following operations on the plaintext.

$$
\begin{aligned}
&\ 0000\ 0000\ 0001 \mathrel{\widehat{=}} 2^{0\cdot 4} \mathrel{\widehat{=}} \text{Vote for Alice \#0}\\
&+\ 0000\ 0000\ 0001 \mathrel{\widehat{=}} 2^{0\cdot 4} \mathrel{\widehat{=}} \text{Vote for Alice \#0}\\
&+\ 0000\ 0001\ 0000 \mathrel{\widehat{=}} 2^{1\cdot 4} \mathrel{\widehat{=}} \text{Vote for Bob \#1}\\
&+\ 0000\ 0001\ 0000 \mathrel{\widehat{=}} 2^{1\cdot 4} \mathrel{\widehat{=}} \text{Vote for Bob \#1}\\
&+\ 0000\ 0001\ 0000 \mathrel{\widehat{=}} 2^{1\cdot 4} \mathrel{\widehat{=}} \text{Vote for Bob \#1}\\
&+\ 0001\ 0000\ 0000 \mathrel{\widehat{=}} 2^{2\cdot 4} \mathrel{\widehat{=}} \text{Vote for Carl \#2}\\
\hline
&= \underbrace{0001}_{1}\underbrace{0011}_{3}\underbrace{0010}_{2}\\
&\mathrel{\widehat{=}} 1\cdot 2^{2\cdot 4} + 3\cdot 2^{1\cdot 4} + 2\cdot 2^{0\cdot 4}
\end{aligned}
$$

The number of votes cast for each candidate i can be revealed by evaluating the corresponding bitspace of the plaintext.

Ballot Preparation The ballot consists of two halves which can be separated by a perforation down the middle. The left hand side contains the permutated

[17] This can be achieved by choosing a random permutation that leads to one shift value per candidate instead of one shift value for the entire list.

candidate list. The right hand side contains a box against each voting option, where the voters can mark their choice and the respective encoded counter. To preserve privacy, each counter is encrypted using an additively homomorphic encryption scheme Enc. Alternatively, the encrypted information can be represented by a 2D barcode or an ID that links to corresponding information published[18]. Furthermore, the right hand side contains an audit strip that can be used to check the well-formedness of the ballot (see Figure 7.2). Each ballot

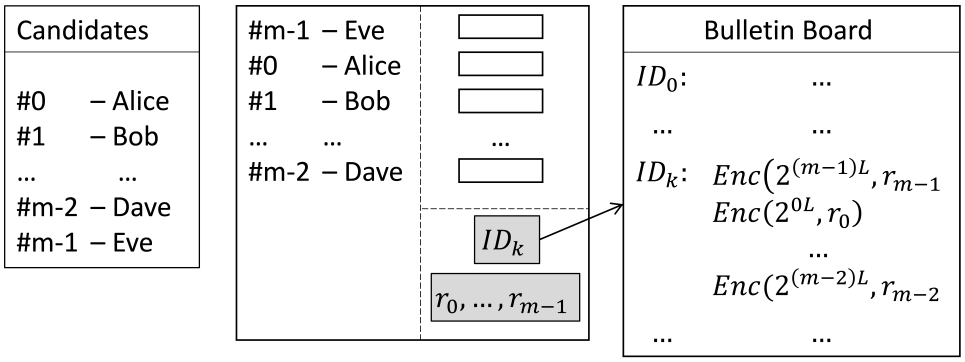


Figure 7.2: Initial candidate ordering (left), ballot paper (middle), and data published on the bulletin board (right).

is generated by performing the following steps:

1. For each candidate $i \in [0, m-1]$ an encrypted counter is generated by computing $\mathsf{Enc}(2^{iL})$.
2. A random shift value s is chosen and the candidate list and the list of encoded counters are permuted.
3. An ID to the ballot is generated and published together with the encrypted counters on the bulletin board.
4. The ballot is printed using the shifted candidate list, the ID, and the random values used to encrypt the counters.

Example. Assume the initial candidate ordering is (Alice #0, Bob #1, Carl #2). For each candidate a counter is computed, that is $\mathsf{Enc}(2^{0L})$ for Alice, $\mathsf{Enc}(2^{1L})$ for Bob, and $\mathsf{Enc}(2^{2L})$ for Carl. The authorities choose a shift value $s = 2$ and generate the permuted candidate list (Bob, Carl, Alice) and the list of encoded counters $(\mathsf{Enc}(2^{1L}), \mathsf{Enc}(2^{2L}), \mathsf{Enc}(2^{0L}))$.

[18]To provide receipt-freeness, the link should be hidden by a scratch field.

Vote Tallying After the polling stations closed, the authorities aggregate all votes cast into one single encrypted tally. This is performed by publicly multiplying all ciphertexts associated to the marked bubbles leading to

$$\mathsf{Enc}(a2^{0L} + b2^{1L} + \ldots + c2^{(m-1)L}),$$

where a denotes the number of votes cast for candidate #0, b the number of votes for candidate #1, and c the number of votes for candidate $\#m-1$ respectively. Finally, the election outcome is decrypted by the key trustees and published.

Example. Assume $L = 4$ bits are used per candidate and 3 votes were cast. Two for Alice, $\mathsf{Enc}(2^{0L}, r)$ and $\mathsf{Enc}(2^{0L}, r')$, and one for Carl $\mathsf{Enc}(2^{2L}, r'')$. Afterwards, all ciphertexts are multiplied obtaining $\mathsf{Enc}(2 \cdot 2^{0 \cdot 4} + 1 \cdot 2^{2 \cdot 4}, r + r' + r'')$. Note that the election outcome can easily be obtained by evaluating the binary representation of the plaintext, that is $(\underbrace{0001}_{1}\underbrace{0000}_{0}\underbrace{0010}_{2})$.

Scratch & Vote Providing Everlasting Privacy Towards the Public

The standard Scratch & Vote voting system has two vulnerabilities. First, it does not provide everlasting privacy, and, second, the ballots are generated by one authority only allowing the "printer knowledge" attack (see Section 3.1.1). In this section we discuss possible improvements with respect to the encoding and ballot preparation procedure. Details to the other election processes, for instance, the ballot printing, vote casting, and auditing procedure, can be found in Section 6.1.

Everlasting Privacy Scratch & Vote can easily be improved to provide everlasting privacy towards the public by replacing the published encrypted information with unconditionally hiding commitments. More precisely, the counter for each candidate $i \in K$ is encoded using a randomly chosen decommiment value t_i by computing $\mathsf{Com}(2^{iL}, t_i)$. Afterwards, the opening values are encrypted using the encryption algorithm $\mathsf{Enc}_{\mathcal{M}}$ and $\mathsf{Enc}_{\mathcal{R}}$. A definition of these cryptographic primitives can be found in Section 2.2. Note that Com and $\mathsf{Enc}_{\mathcal{M}}$ must be additively homomorphic with respect to the message space.

It follows that a vote for candidate i is represented by the following three components.

$$\langle u_i, v_i, w_i \rangle = \langle \mathsf{Com}(2^{iL}, t_i), \mathsf{Enc}_{\mathcal{M}}(2^{iL}), \mathsf{Enc}_{\mathcal{R}}(t_i) \rangle$$

For each ballot, the commitments for each candidate u_i are published while the corresponding encrypted opening values v_i and w_i are privately stored in a key server. The auditing strip contains a scratch field that hides the decommitment values $t_0, \ldots, t_{m-1}$. This allows each voter to audit the well-formedness of some ballots before voting.

After the vote casting process, the commitments associated to the marked positions are publicly multiplied, leading to

$$u^* = \mathsf{Com}(a2^{0L} + b2^{1L} + \ldots + c2^{(m-1)L}, t^*),$$

where, similar to the classic approach, a constitutes the number of votes for candidate #0, b for #1, and c for $\#m-1$.

The system privately multiplies the corresponding encrypted opening values and obtains

$$v^* = \mathsf{Enc}_{\mathcal{M}}(a2^{0L} + b2^{1L} + \ldots + c2^{(m-1)L})$$
$$w^* = \mathsf{Enc}_{\mathcal{R}}(t^*)$$

By opening the commitment using the decrypted opening values $a2^{0L} + b2^{1L} + \ldots + c2^{(m-1)L}$ and t^*, the system can prove that the election result was computed correctly.

Printer Knowledge Both versions of Scratch & Vote, the standard and the improved scheme, can easily be adapted such that ballots can be generated and printed in distributed fashion. Note that for each additively homomorphic encryption scheme $\mathsf{Enc}(s)$ it holds that

$$\mathsf{Enc}(s)^t = \mathsf{Enc}(t \cdot s).$$

Thus, assuming the participation of several clerks, a ballot can be generated in distributed fashion by performing the following steps.

1. Clerk 1 generates a counter for each candidate by computing $\{2^0, 2^L, \ldots, 2^{(m-1)L}\}$. Afterwards, he or she chooses a random seed s_1 and shuffles the list of candidates and counters correspondingly,

   $$\{2^{(0-s_1)\cdot L}, 2^{(1-s_1)\cdot L}, \ldots, 2^{(m-1-s_1)\cdot L}\}.$$

   Finally, clerk 1 encodes each entry $i \in [0, m-1]$

   $$\langle u_i, v_i, w_i \rangle = \langle \mathsf{Com}(2^{(i-s_1)L}, t_i), \mathsf{Enc}_{\mathcal{M}}(2^{(i-s_1)L}), \mathsf{Enc}_{\mathcal{R}}(t_i) \rangle,$$

   where t_i is a randomly chosen decommitment value and sends this information together with the encrypted data needed for printing (compare to Section 6.1.2) to the next clerk.

2. Clerk 2 chooses a second seed s_2' at random. Afterwards, he or she updates the encrypted data for printing as described in Section 6.1.2. Finally, the clerk updates the counters by computing for each entry $i \in [1, m-1]$

$$\begin{aligned}\langle u_i', v_i', w_i' \rangle &= \langle u_i^{2^{-s_2' L}}, v_i^{2^{-s_2' L}}, w_i^{2^{-s_2' L}} \rangle \\ &= \langle \mathsf{Com}(2^{(i-s_1)L}, t_i)^{2^{-s_2' L}}, \mathsf{Enc}_{\mathcal{M}}(2^{(i-s_1)L})^{2^{-s_2' L}}, \mathsf{Enc}_{\mathcal{R}}(t_i)^{2^{-s_2' L}} \rangle \\ &= \langle \mathsf{Com}(2^{(i-s_1-s_2')\cdot L}, 2^{-s_2' L} \cdot t_i), \mathsf{Enc}_{\mathcal{M}}(2^{(i-s_1-s_2')\cdot L}), \mathsf{Enc}_{\mathcal{R}}(2^{-s_2' L} \cdot t_i) \rangle \\ &= \langle \mathsf{Com}(2^{(i-s_2)\cdot L}, t_i'), \mathsf{Enc}_{\mathcal{M}}(2^{(i-s_2)\cdot L}), \mathsf{Enc}_{\mathcal{R}}(t_i') \rangle\end{aligned}$$

3. All clerks process the data the same way. The final output is a permuted list of candidates and a permuted set of counters where the knowledge of the shift value used is shared among a set of clerks.

7.2.2 Properties

With respect to **correctness**, **individual verifiability**, **universal verifiability**, and **robustness** this voting system provides the same properties as shown for the homomorphic tallying protocol in Section 7.1.4. Furthermore, it provides **receipt-freeness** if the ID printed on the ballot is hidden by a scratch field. With respect to **computational privacy** this scheme improves the standard Scratch & Vote voting system since it provides a distributed printing and ballot generation process preventing the "authority knowledge" attack.

Furthermore, it also provides **everlasting privacy towards the public** because only unconditionally hiding commitments are published. However, the encrypted opening values are processed in the back end which is why the same organizational measures, like for the improved Prêt à Voter scheme, are needed to provide **everlasting privacy towards the authorities**. While the encoding of votes used in mixing based voting schemes allows for secret sharing, this cannot be applied to homomorphic counters. An authority is able to permute the shift value encrypted by another authority, as explained for the distributed printing process. However, this is not possible if both shares that constitute the shift values are available in encrypted form only. On the one hand an additive encoding scheme is needed because in order to increase the counters they must be added, i.e.,

$$\mathsf{Enc}(a2^{s_1}) \cdot \mathsf{Enc}(2^{s_1}) = \mathsf{Enc}((a+1) \cdot 2^{s_1})$$

On the other hand the shift values of two counters can be aggregated only if an operation on the encryptions leads to a multiplication of the secret, i.e.,

$$\mathsf{Enc}(2^{i-s_1-s_2}) = \mathsf{Enc}(2^{i-s_1}) \cdot \mathsf{Enc}(2^{-s_2})$$

But since all known primitives are either additively homomorphic or multiplicative homomorphic another approach, for instance, based on the protocol by [CFSY96] or by using some publicly verifiable multiparty computation techniques [CDN01, BDO14] is needed. However, finding a layout and protocol that allows for secret sharing is out of scope for this work and will be left for future work.

8 Towards a Voting Scheme Providing Non-Personalized Receipts

To provide verifiability, in many voting systems a receipt is handed out to the voters containing their vote in encoded form. However, to reveal the votes cast, some decoding information is needed. Even though the access key to this information is shared among several persons in authority, the voter has to trust in a subset of officials. Assumptions frequently made are, for instance, that at least one mix out of n mixes is honest when using a mix-net for anonymization, or that a subset of key holders keep their key shares secret. Thus, these schemes assume that voters are willing to put trust in authorities.

In 2006, two new voting systems were proposed: a verifiable variant of the Farnel scheme [ACG07] and Twist [RS07]. Both schemes introduce a different approach for issuing receipts: a voter gets a receipt not of his or her own ballot, but of other ballots already cast. As a result, voter privacy cannot be violated during the anonymization and tallying procedure, even if all authorities collaborate. Unfortunately, the ballot box used in these schemes have some drawbacks regarding correctness and privacy. The decision which ballots are selected for a receipt depends on the order in which the votes were cast. Especially the votes that have been submitted later have a lower chance to be chosen. Usually only a subset of the votes cast is printed on receipts and verified by voters. Furthermore, since each receipt can contain only votes that have been submitted by predecessors, an attacker can get additional information by collecting all receipts.

Thus, first we describe the improved verifiable Farnel scheme proposed by Araújo and Ryan in 2008 [AR08] and summarize the vulnerabilities of the used receipt generation process. Second, we propose a novel electronic ballot box that generates receipts in a universal verifiable and efficient manner. Our solution improves the classic receipt generation process of non-personalized receipts with respect to both privacy and correctness.

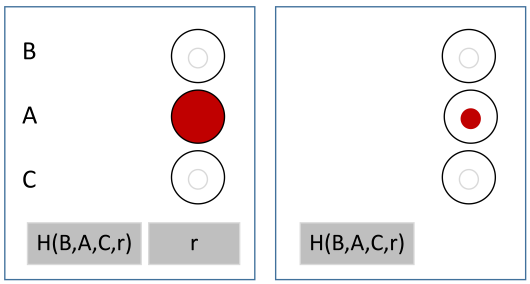


Figure 8.1: Ballot paper layout of the verifiable Farnel voting system. Top layer (*left*) and bottom layer (*right*) generated with hash function H and random value r.

8.1 The Verifiable Farnel Voting Scheme

In this section we describe and analyze the Farnel voting scheme proposed by Araújo and Ryan [AR08].

8.1.1 Ballot Form

The ballot layout used is composed of two layers. The top layer shows the candidate names in random order and allows choosing them by marking the corresponding bubble. In addition, it also includes a hash to the random candidate order and the respective opening values, both hidden under scratch fields. If the voter fills out a ballot, the position is recorded by the bottom layer which only contains a scratch field hiding the hash value. Thus, the top layer shows the marked candidates while the bottom layer reflects the same vote in encoded form (see Fig. 8.1). The simultaneously completion of both layers can be provided by carbon paper or wholes punched in the top layer.

8.1.2 Election Preparation

Prior to the vote casting process, a predefined number of ballot papers is audited. The authorities reveal the hash and opening values and check their consistency with respect to the used candidate order. Afterwards, the audited ballots are discarded. Following, a predefined number, *init*, of initial votes is cast. The ballots should be filled out such that each voting option gets the same amount of initial votes to not give advantage to a particular candidate. The filled out top layers are cast in a conventional ballot box, while the bottom layers are submitted to the Farnel Box.

8.1.3 Vote Casting Process

The voter receives a ballot paper that he or she can either use to audit or to vote. After the voter filled out the ballot, he or she puts it into a special envelope which just shows the scratch surfaces but not the marked bubbles. Subsequently, he or she leaves the secret polling booth and the poll workers verify that the layers have not been separated and that the scratch surfaces are still intact. If the ballot was filled out properly, the voter cast the top layer into the conventional ballot box. Furthermore, he or she submits the bottom layer to the Farnel Box which removes the scratch field, shuffles its content, and outputs a receipt that contains a subset of votes cast by predecessors. Note that the votes printed on the receipt can, but does not necessarily, include the vote cast by the voter him or herself.

8.1.4 Tallying Process

To determine the election result, both ballot boxes, the conventional box and the Farnel Box, publish their content. More precisely, the scratch fields of all ballots cast in the conventional box are revealed and it is checked whether the commitments and opening values match the corresponding candidate list and whether the same position has been marked on both layers. From the information published, each voter can verify that the codes printed on his or her receipt are included in the tally. Finally, the votes cast during the initialization process are subtracted and the remaining votes are publicly tallied.

8.1.5 Drawbacks and Vulnerabilities

Compared to voting systems using personalized receipts, this scheme has some drawbacks regarding correctness and privacy.

Correctness One drawback of the verifiable Farnel scheme is that the probability of an encrypted vote to appear on a receipt depends on the time when the vote was cast. Thus, votes that were submitted at the end of the vote casting process have less chance being verified by other voters and can be replaced more easily (see [Ara08]).

Privacy The fact that early cast votes have a higher chance of being selected than the votes cast by the last voters leads to another drawback of the verifiable Farnel scheme. The frequency of votes appearing on receipts might allow conclusions regarding the time when the votes were submitted (see [Ara08]). Having, for instance,

K voters and correspondingly K receipts, the vote cast by the first voter may appear on K receipts while the vote cast by the last voter can be selected for the last receipt only.

Another vulnerability is that this scheme is prone to vote clustering (see Section 3.1.2). The vote of the last voter, for instance, cannot be printed on receipts before he or she submitted his or her vote. Thus, this voter can be mapped to a subset of votes cast: all that have not been printed before the last voter cast his or her vote. During the tallying process these votes become public and if this set does not include at least one vote for each candidate, a coercer knows whom the last voter did not vote for. More precisely, suppose we run an election with four candidates: Alice, Bob, Carl, and Dave. The last voter votes and receives a ballot containing five hashes. Two hashes link to votes for Alice and three to votes for Carl. In addition a subset of hashes contained in the Farnel Box has not been printed on receipts. If this subset contains only votes for Alice, Bob, and Carl, an attacker knows that the last voter did not vote for Dave. Note that if a coercer wants to verify whether bribed voters followed the instructions, it might be sufficient to know whom they did not vote for.

This weakness has gone unnoticed so far and is not just a vulnerability with respect to the privacy of the last voter. Assume the following situation: We have K voters and voter number $i \leq K$ received a receipt containing his or her own vote. In addition all votes, 1 to $i-1$, that have been cast before and all votes cast during the initialization process were chosen for at least one receipt. In this case we have an "unintended intermediate result". All votes contained in the Farnel Box were printed. It follows that all initial votes and all votes cast by voter 1 to i are in this subset, determined by the first i receipts. After the vote casting process, the receipts are publicly decrypted. Therefore, an attacker can determine the intermediate result after the first i voters cast their vote. He or she counts the values associated to the hashes printed on the first i receipts and subtracts the votes cast in the initialization phase. If this set does not include a vote for Alice the attacker knows that none of the first i voters cast a vote for her. Note that this also allows conclusions regarding the votes cast by the last $K-i$ voters, for instance, if none of them voted for Bob.

8.2 Technical Details of an Electronic Ballot Box

To address the issues identified in the last section, we introduce a novel electronic ballot box that generates receipts by weighted random sampling. Our solution can be applied to voting systems that use a homomorphic encryption scheme to encrypt the

votes cast. Regarding correctness this approach provides a similar probability for all votes cast being printed on receipts. Furthermore, the weights of votes submitted in the initialization phase range between a low value ϵ and 1 and are equally distributed for all candidates. Thus, not all initial votes are selected at the beginning of the vote casting process. Furthermore, some of them are not printed on receipts handed out to voters, which is why the set of votes that do not show up on receipts contain votes for all candidates. This addresses the privacy vulnerabilities described in Section 8.1.5 and allows voters to pass their receipt to helping organizations without risking that an attacker is able to collect enough information to violate voter privacy.

In this section the technical details of the setup, vote casting, closing, and tallying process are described.

8.2.1 Encoding of Votes

The input to the electronic ballot box are votes encrypted with a homomorphic public-key encryption scheme as described in Section 2.2.1. This can, for instance, be achieved by using the standard Prêt à Voter ballot which allows generating an encrypted vote by filling out a paper ballot[19].

8.2.2 Setup

In the following section we will describe the setup process for m candidates and K ballots. For each position on each ballot an encryption is generated that reflects a vote for the corresponding candidate. In addition to the stack of printed ballots, each polling station receives a USB stick or CD containing these $m \cdot K$ ciphertexts. Then the device initializes itself by performing the following steps:

1. An input table is generated which contains all possible voting decisions for each ballot in encrypted form

$$\{Enc(v_{i,j}, t_{i,j})\}_{1 \leq i \leq m, 1 \leq j \leq K},$$

where $j \in [1, K]$ denotes the ballot, $i \in [1, m]$ the position, and $t_{i,j}$ the random value[20]. Each row corresponds to one position on one ballot paper and records the encrypted vote and a flag that is set when the position has been marked by a voter.

[19] In this case the used public-key encryption scheme must be additive homomorphic. The correct encryption can be verified by revealing the randomness used as "opening value", similar to the Benaloh Challenge [Ben07].

[20] Note that the paper ballots must be shuffled to prevent that voter j fills our ballot j.

2. An output table is generated, similar to the mixing process described in Section 4.1, containing
   - the reencrypted and shuffled ciphertexts of the input table,

     $$\pi\{Enc(v_{i,j}, t'_{i,j})\}_{1\leq i\leq m, 1\leq j\leq K} \text{ where}$$

     $$Enc(v_{i,j}, t'_{i,j}) = Enc(v_{i,j}, t_{i,j}) \cdot Enc(0),$$

   - a flag that is set when the corresponding position on the scanned ballot has been marked and
   - a column that records a weight for each flagged entry initialized by 0 (see Fig. 8.2).

3. Both tables, the input and the output table, are published together with a proof of correct shuffling (see Section 2.2.3) on the public accessible bulletin board. It follows that any interested party can verify that the set of anonymized encrypted votes,

   $$\pi\{Enc(v_{i,j}, t'_{i,j})\}_{1\leq i\leq m, 1\leq j\leq K}$$

   is a correct reencryption and shuffling of

   $$\{Enc(v_{i,j}, t_{i,j})\}_{1\leq i\leq m, 1\leq j\leq K}$$

   such that both sets encrypt the same batch of voting decisions

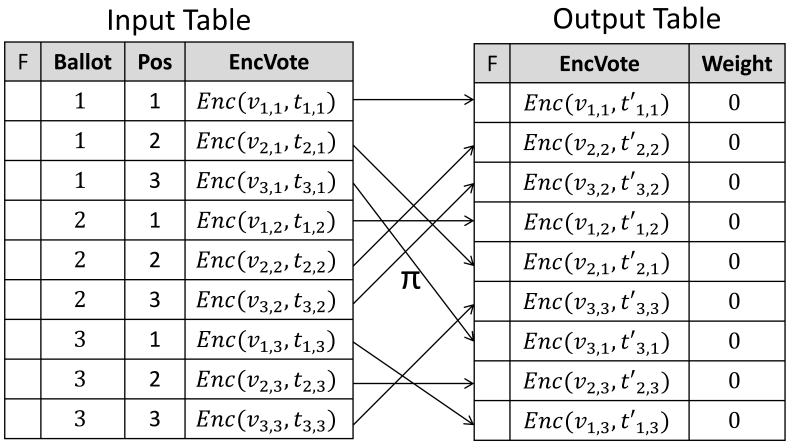
   $$\{v_{i,j}\}_{1\leq i\leq N, 1\leq j\leq K}.$$

   Note that, to provide voter privacy, the electronic ballot box keeps the permutation π and the values used to reencrypt the input table secret.

4. A predefined number of initial votes are cast. Having ballots with m candidates, then the weight, w_i, for initial vote i is calculated by a function

   $$f(i, m) \in [\epsilon, 1]$$

   which returns a weight that ranges between $\epsilon \geq 0$ and 1. This function must ensure equal probability of all candidates to be chosen during the receipt generation process.



F	Ballot	Pos	EncVote
	1	1	$Enc(v_{1,1}, t_{1,1})$
	1	2	$Enc(v_{2,1}, t_{2,1})$
	1	3	$Enc(v_{3,1}, t_{3,1})$
	2	1	$Enc(v_{1,2}, t_{1,2})$
	2	2	$Enc(v_{2,2}, t_{2,2})$
	2	3	$Enc(v_{3,2}, t_{3,2})$
	3	1	$Enc(v_{1,3}, t_{1,3})$
	3	2	$Enc(v_{2,3}, t_{2,3})$
	3	3	$Enc(v_{3,3}, t_{3,3})$

F	EncVote	Weight
	$Enc(v_{1,1}, t'_{1,1})$	0
	$Enc(v_{2,2}, t'_{2,2})$	0
	$Enc(v_{3,2}, t'_{3,2})$	0
	$Enc(v_{1,2}, t'_{1,2})$	0
	$Enc(v_{2,1}, t'_{2,1})$	0
	$Enc(v_{3,3}, t'_{3,3})$	0
	$Enc(v_{3,1}, t'_{3,1})$	0
	$Enc(v_{2,3}, t'_{2,3})$	0
	$Enc(v_{1,3}, t'_{1,3})$	0

Figure 8.2: The input and output table for three ballots and three candidates after the setup phase.

8.2.3 Vote Recording

The filled out bottom layer is scanned and the ciphertext is sent to the electronic ballot box.

1. The electronic ballot box records the vote cast by setting the corresponding flags in both tables[21].

2. The corresponding weight is set to a predefined value w, where $0 < w \leq 1$ (see Fig. 8.3).

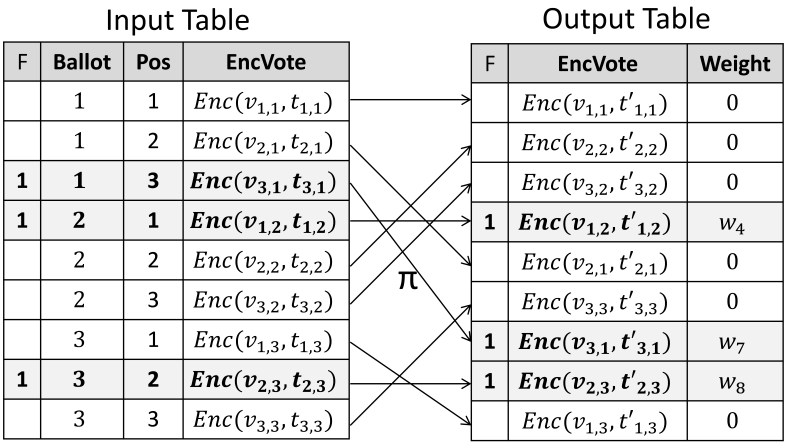
3. The electronic ballot box generates the data for the receipt using the acceptance/rejection method [Rub81]:

   a) It randomly selects a row j with weight w_j out of all flagged entries.

   b) Then a value p between 0 and 1 is chosen uniformly at random. If $p < w_j$ the vote is marked to be printed on the receipt and the weight is reduced by factor q by calculating $w_j := w_j \cdot \frac{1}{q}$. Otherwise the row is not marked.

   c) Step 3(a) and 3(b) are repeated until l entries have been found, where l denotes the number of votes printed on each receipt.

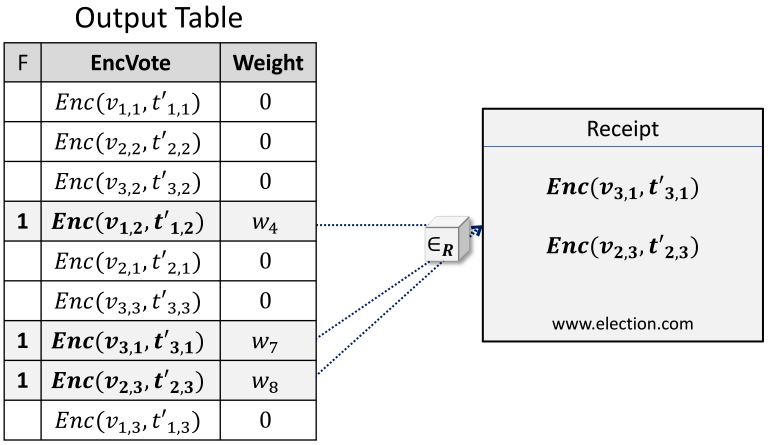
[21] Due to the proposed shuffling procedure the information which candidates have been marked together on one ballot is lost. However, if needed, the process can easily be adapted, for instance, by shuffling tuples that represent the entire ballot. The candidates chosen on each ballots can still be marked by setting flags, which prevents an enlargement of the tables stored in the electronic ballot box.



F	Ballot	Pos	EncVote
	1	1	$Enc(v_{1,1}, t_{1,1})$
	1	2	$Enc(v_{2,1}, t_{2,1})$
1	**1**	**3**	$\boldsymbol{Enc(v_{3,1}, t_{3,1})}$
1	**2**	**1**	$\boldsymbol{Enc(v_{1,2}, t_{1,2})}$
	2	2	$Enc(v_{2,2}, t_{2,2})$
	2	3	$Enc(v_{3,2}, t_{3,2})$
	3	1	$Enc(v_{1,3}, t_{1,3})$
1	**3**	**2**	$\boldsymbol{Enc(v_{2,3}, t_{2,3})}$
	3	3	$Enc(v_{3,3}, t_{3,3})$

F	EncVote	Weight
	$Enc(v_{1,1}, t'_{1,1})$	0
	$Enc(v_{2,2}, t'_{2,2})$	0
	$Enc(v_{3,2}, t'_{3,2})$	0
1	$\boldsymbol{Enc(v_{1,2}, t'_{1,2})}$	w_4
	$Enc(v_{2,1}, t'_{2,1})$	0
	$Enc(v_{3,3}, t'_{3,3})$	0
1	$\boldsymbol{Enc(v_{3,1}, t'_{3,1})}$	w_7
1	$\boldsymbol{Enc(v_{2,3}, t'_{2,3})}$	w_8
	$Enc(v_{1,3}, t'_{1,3})$	0

Figure 8.3: The input and output table after three votes were cast.



F	EncVote	Weight
	$Enc(v_{1,1}, t'_{1,1})$	0
	$Enc(v_{2,2}, t'_{2,2})$	0
	$Enc(v_{3,2}, t'_{3,2})$	0
1	$\boldsymbol{Enc(v_{1,2}, t'_{1,2})}$	w_4
	$Enc(v_{2,1}, t'_{2,1})$	0
	$Enc(v_{3,3}, t'_{3,3})$	0
1	$\boldsymbol{Enc(v_{3,1}, t'_{3,1})}$	w_7
1	$\boldsymbol{Enc(v_{2,3}, t'_{2,3})}$	w_8
	$Enc(v_{1,3}, t'_{1,3})$	0

Figure 8.4: Receipt generation

4. The electronic ballot box sends the set of l anonymous votes to the printer that generates the receipt (see Fig. 8.4).

8.2.4 Closing

After the vote casting process, the correct functioning of the electronic ballot box must be verified and all information have to be deleted.

1. After the last voter cast his or her vote, a batch of receipts is printed that is verified by the authorities and/or helping organizations.

2. The electronic ballot box publishes the input and output table on the bulletin board. The set of recorded anonymized votes is

$$\{Enc(v_l)\}_{l=1}^{L} := \pi\{Enc(v_{i,j}, t'_{i,j})\}_{(i,j)\in G},$$

where $L = |G|$ is the number of votes cast and G consists of all tuples (i, j) where position i has been marked on ballot j.

3. The electronic ballot box publicly proves that the subset of votes flagged in the output table

$$\pi\{Enc(v_{i,j}, t'_{i,j})\}_{(i,j)\in G}$$

is a correct reencryption and mixing of the set of scanned inputs, marked in the input table

$$\{Enc(v_{i,j}, t_{i,j})\}_{(i,j)\in G}.$$

4. As in the Farnel scheme described in Section 8.1, the poll workers publish the content of the electronic and classic ballot box.

5. If both sets are consistent and the proofs of correct shuffling hold, then all information stored in the electronic ballot box is deleted.

Assuming that all information recorded by the electronic devices, such as the scanner and the electronic ballot box, is irretrievably erased the link between voters and their vote is destroyed.

8.2.5 Tallying

To tally the votes cast, the key holders jointly decrypt the published votes. Subsequently, the amount of votes cast during the initialization step is subtracted, followed by publicly determining the election result.

8.2.6 Organizational and Technical Security Policies

The whole procedure, from setting up the electronic ballot box to verifying and deleting the stored information, should be carried out in public. This allows any interested party to observe that no data stored in the device is leaked. In addition, the electronic ballot box should be protected from unauthorized access by implementing an access control mechanism, where the access key is shared among several persons in authority[22]. Authorized poll workers should be able to call functions and

[22] A hardware security model, for instance, could be used to provide these security requirements.

insert challenges for the verification process but they must not have access to the permutation and random values used. Furthermore, all hardware components, for instance, the electronic ballot box, scanners, and printers, should not contain any permanent memory and must be protected against side channel attacks.

8.3 Implementation and Evaluation

In this section we summarize and discuss the results of our prototypical realization with respect to **correctness**, **verifiability**, **privacy**, and **efficiency**. We only implemented the receipt generation process since for the setup, shuffling, and verification process standard mix-net software can be used. Furthermore, we do not discuss the requirements with respect to the ballot preparation and vote casting process, like **receipt-freeness**, because this depends on the voting system used.

8.3.1 Correctness and Verifiability

Araújo performed several experiments to identify optimal parameters for the Farnel Box [Ara08]. He used a constant number of 500 voters who cast one vote each, a receipt size that varies between $l = 1$ and $l = 5$, and an initialization set, *init*, of 100, 500, 800, and 1000 votes. To simplify the comparison, we used the same parameters and set the number of candidate to 5. To get significant results, each test was performed 10,000 times[23].

The simulations by Araújo showed that the highest probability to detect a modified vote is 83.41%, which is achieved for an initialization set of 100 votes and 5 votes printed on each receipt [Ara08, Fig. 3.5(b)]. Furthermore, his simulation showed that the votes cast by the first 138 voters have a 99% chance of being selected for at least one receipt, afterwards the probability falls exponentially.

To be able to compare our results, we first have to identify the optimal parameters for weighted random selection for an initialization set of 100 votes. To determine this we ran several tests and obtained the best results for the following values:

- Cast votes are initialized with weight $w = 1$.
- Votes chosen for a receipt are adapted with a reduction factor of $q = 2$.
- The minimum weight is $\epsilon = 0.0002$.

[23] We run tests with 10,000 iterations two times for the same parameters and the results only slightly differed.

- The weights for the votes cast in the initialization phase are calculated with the function

$$f(1, m) = \min(1, \epsilon \cdot p^{\lfloor \frac{i}{m} \rfloor}),$$

where $m = 5$ is the number of candidates per ballot, $p = 3$ a second reduction factor, and $i \in [1, 100]$ the index of the initial vote.

Note that by using a reduction factor p in the function $f(1, m)$ we simulate the state of the electronic ballot box after i votes have been cast and approximately i receipts have been generated. The weight is set in a way that all candidates have the same chance to be selected (see Tab. 8.1).

Table 8.1: Weights for the votes cast in the initialization phase

Initial Vote i	Candidate	Weight w_i
1	1	0.0002
2	2	0.0002
3	3	0.0002
4	4	0.0002
5	5	0.0002
5	1	0.0004
6	2	0.0004
...	...	...
11	1	0.0008
...	...	...
61	1	0.8192
...	...	...
66	1	1
...	...	...
100	5	1

With weighted random selection votes that have not been printed on receipts so far are preferentially selected. Our simulations show that, on average, the votes of the first 452 voters appear with a probability of 99% using the same parameters as Araújo. Only 30.1085 votes cast by voters were not printed on receipts. Thus, weighted random selection also increases the probability to detect modified votes to 93.9783%. Note that for correctness we do not take into account that after the vote casting process has ended, additional receipts are printed and verified, since this is not carried out by voters. Nevertheless, auditors will do some additional tests and thus, in practice, the probability of detecting manipulations is higher.

8.3.2 Privacy

Our proposal is not only an improvement with respect to correctness, but also with respect to privacy. Using weighted random selection early cast votes do not appear more often on receipts than votes that were cast later. This prevents statistical attacks on receipts. Furthermore, we ran experiments to check whether

1. an "unintended intermediate result" can be computed; and
2. voters can be mapped to a subset of votes cast.

We ran 10,000 simulations with the parameters determined for 5 candidates and the event of an "unintended intermediate result" never occurred. Furthermore, on the average 3.955 initial votes per candidate were not printed on receipts. This sufficiently hides the voting decisions cast by the last voters. Furthermore, tests carried out with a prototypical implementation also showed that the number of initial votes can be lowered with respect to the number of candidates while still getting satisfying results regarding privacy.

However, for voter privacy it is essential that the data stored in the electronic ballot box is not leaked. Thus, the device should be observed during the whole duration of use. This is possible because the ballot box is set up and ran in the polling station. After the vote casting process, before the votes are decrypted, all information stored on the electronic ballot box is publicly deleted. Under the assumption that the tables are irrevocably erased, the link between the votes contained in the tally and the scanned information is destroyed and voter privacy is ensured. This allows a higher level of privacy compared to voting systems with personalized receipts because the voters do not have to put trust in officials keeping their key shares secret.

8.3.3 Efficiency

The large number of initial votes that were not printed on receipts allowed us to reduce the size of the initialization set needed to ensure voter privacy. We ran several tests to determine the number of initial votes. All tests measured correctness, i.e., the probability of detecting a modified vote, the number of initial votes per candidate that remain in the electronic ballot box, and checked whether an "unintended intermediate result" can be computed. We summarized our results for 5, 10, and 20 candidates in Table 8.2.

For ballots that contain less than 10 candidates we could reduce the number of votes that have to be cast in the initialization phase. Such a smaller number of

Table 8.2: Size of the initialization set for elections with 5, 10, and 20 candidates having 500 voters and 5 votes per receipt

Candidates m	5	10	20
Initial Votes *init*	50	100	140
Weight w	1	1	1
Reduction Factor q	2	2	2
Initialization Function $f(1,m)$	$\min(1, 0.002 \cdot 3^{\lfloor \frac{i}{5} \rfloor})$	$\min(1, 0.002 \cdot 4^{\lfloor \frac{i}{10} \rfloor})$	$\min(1, 0.002 \cdot 4^{\lfloor \frac{i}{20} \rfloor})$
Correctness	96.8538	94.7801	91.3626
Unpublished Initial Votes per Candidate	1.4677	1.5849	1.625
Unpublished Votes Cast	15.7310	26.0995	43.1869

initial votes is an improvement with respect to two aspects. First, this means less work for officials and, second, a small initialization set increases the probability that all votes cast are chosen at least once. For 5 candidates the probability of detecting a modification could be increased from 93.9783% to 96.8538%. Furthermore, on the average, the number of votes cast in the initialization step that were not printed on receipts is 1.4677. This makes it impossible for an attacker to distinguish from the published information which votes were cast by the last voters and which were cast during the initialization phase.

To analyze the efficiency for 500 voters, 20 candidates, and a receipt size of 5, we built a table for 1,500 ballots. The table consists of 30,000 rows, 20 rows per ballot. The execution times for the receipt generation were measured on an Intel(R) Core(TM) 2 Duo CPU T7500 at 2.20GHz. We run 10,000 tests and the average time was 91,254 ns while 836,070 ns was the maximum time needed to select 5 votes for a receipt.

8.3.4 Verifiability

To verify that the votes were processed correctly, we have to check that all votes are **recorded as intended** and **tallied as recorded**.

Recorded as Intended

As in the verifiable Farnel scheme, during the vote casting process, the set of filled out ballots is collected in a conventional ballot box. After the vote casting process

has ended this batch is scanned and compared with the batch of scanned ballots marked by the electronic ballot box. This ensures that the device sets the flags for the correct entries in the input table. Instead of checking the consistency of all ballots, alternatively, spot tests can be carried out.

Subsequently, after the vote casting process has ended, the electronic ballot box publishes the set of votes marked in the input table and the set of anonymized votes printed on receipts. Then it uses a proof of correct shuffling to publicly prove that both sets encrypt the same batch of votes cast. This ensures consistency between the input and output table with respect to the rows marked.

Tallied as Recorded

During the vote casting process, each voter receives a receipt containing a subset of anonymized votes cast by predecessors. This allows him or her to verify that the ciphertexts printed on their receipt are included in the tally. Thus, if the electronic ballot box during the vote casting process or authorities afterwards modified, deleted, or replaced votes this will be detected under the assumption that a sufficient number of voters check their receipt.

The size of the initialization set, the number of votes printed on each receipt and the number of receipts printed after the last voter cast his or her vote is chosen so that each vote cast is printed on at least one receipt with high probability.

Robustness

The system provides robustness because the filled-out top layers are collected in a conventional ballot box. This allows recounting in case of a malfunction or breakdown.

9 Conclusion and Future Work

In this work we analyzed and improved verifiable voting systems with respect to privacy. First, we introduced a generic verification process that allows verifying the correctness of the anonymization process used in mixing based voting systems. In contrast to other efficient approaches, this process prevents that encoded votes can be linked to a subset of possible voting options, even if only one mix acts honestly. Furthermore, our solution is competitive with respect to correctness and efficiency.

Furthermore, we introduced two novel universally verifiable mix-nets: one that provides everlasting privacy with respect to the information published and one that provides everlasting privacy towards both the public and the authorities. A very important property of these mixing protocols is that they can be applied to existing voting schemes without significant changes to the classic implementation. We showed how the first mix-net can be used to introduce everlasting privacy to Prêt à Voter and described how the second protocol improves the Split-Ballot voting system.

Subsequently, we proposed a homomorphic tallying procedure that provides verifiability and everlasting privacy. This protocol can be used to improve poll-site voting systems where the votes are homomorphically tallied, such as Scratch & Vote.

Finally, we analyzed how non-personalized receipts can be generated using the example of the Farnel voting scheme. We introduced an improved electronic ballot box which allows generating receipts in an efficient and universally verifiable manner. Due to a weighted random sampling when choosing the votes for receipts, the classic approach was improved regarding correctness and privacy. The set of votes cast that are printed at least on one receipt was increased, raising the probability of detecting modified votes. Furthermore, by collecting all receipts, an attacker cannot reduce the set of recorded voting options containing the votes cast by the last voters.

Comparison

In this work we improved four voting systems, Prêt à Voter, Split-Ballot, Scratch & Vote, and Farnel with respect to privacy. All these schemes have their advantages and disadvantages. The new version of Prêt à Voter, for instance, provides everlasting privacy towards the public. However, to ensure everlasting privacy towards the authorities certain organizational measures are needed since one malicious computationally unbounded authority is enough to violate voter privacy.

By contrast, Split-Ballot has no single point of failure with respect to everlasting privacy but a more complex vote casting process because the ballot consists of three layers and two mix-nets are used that must be verified by auditors and observers. Furthermore, it is not clear how the ballot layout can be modified to provide an arbitrary candidate order. Thus, the candidate lists can only be shifted cyclicly and privacy is not provided if each voter cast more than one vote. Thus, this scheme is not suitable for all elections.

In Scratch & Vote, the votes are tallied homomorphically. Compared to Prêt à Voter and Split-Ballot, no mix-net needs to be used and verified which is why this scheme demands less work for authorities and voters. However, Scratch & Vote can only be used for elections where the outcome can be determined by homomorphic tallying. If a ballot has certain restrictions, for instance, two choices may not be marked together on one ballot, it must be possible to evaluate the filled-out ballot. More precisely, suppose the ballot papers have a "spoil" option and each voter is allowed to either cast two votes or to spoil the ballot. If he or she marked the bubble associated to "spoil" and selected a further candidate, the vote for the candidate does not count. In this case the votes cannot be tallied homomorphically. To check whether the voters followed this restriction, all ballots must be evaluated prior to the tallying process. Furthermore, similar to Prêt à Voter the improved version of Scratch & Vote does not allow for secret sharing and provides everlasting privacy towards the authorities only under certain organizational measures.

Farnel differs significantly from the other schemes with respect to correctness, verifiability, and privacy. Correctness and verifiability depend on the number of votes cast that appear at least once on a receipt. This puts the last voters at a disadvantage, because their votes have a lower change to be chosen. Furthermore, privacy is provided only if all information recorded by the electronic Farnel box is destroyed before the votes are decrypted and tallied. However, this approach is the first step in the direction of verifiable voting systems where not only the correctness of the tallying process can be verified but also voter privacy becomes observable, similar to a traditional voting system. If no information is leaked during the use

of the Farnel Box in the polling station and if all data stored on the memory is destroyed in public, voters can have confidence that the used voting system protects their privacy. In all other schemes, Prêt à Voter, Split-Ballot, and Scratch & Vote, they must put trust in a subset of key holders.

Which of those systems fits best for an election depends on the electoral system and its legal regulation. In some cases, organizational measures might be not sufficient to preserve everlasting privacy. In other cases, procedural controls are acceptable but the ballot layout should be as simple as possible. Furthermore, in some elections each voter selects only one candidate while in other elections he or she has more than one vote or several races on one ballot. Also with respect to verifiability the requirements may differ. While some legal regulations might enforce that each voter should be allowed to verify that his or her own vote is contained in the tally, other regulations may prefer that the link between a voter and his or her vote is broken during the vote casting process. Thus, the decision which voting system should be used needs to be discussed for each election individually.

Future Work

Before an electronic voting system can be applied to a particular legally binding election, further research with respect to all electoral principles is necessary. Furthermore, it needs to be analyzed to which extent the scheme meets other requirements of the electoral system, like scalability, usability, efficiency, and election versatility. Ballots using a random candidate ordering, for instance, are not well-suited for elections with a large number of candidates.

Especially with respect to impersonalized receipts further research is necessary. The receipt generation process presented in this work has been only experimentally tested. To determine optimal parameters for different elections, another more analytical analysis should follow. But also other schemes parented here need to be further elaborated. The layout of Split-Ballot, for instance, should be improved such that it provides arbitrary candidate lists and not only shifted ones. Furthermore, a scan-based voting system that allows for homomorphic tallying and secret sharing should be developed. Using Scratch & Vote everlasting privacy towards the authorities can be provided by organizational measures only.

The contribution of this work with respect to everlasting privacy, the mixing and tallying process, are generic in the sense that they can be applied to other protocols that use homomorphic tallying and/or mix-nets. With respect to online voting, for instance, everlasting privacy has so far only be introduced to Helios. But there

are other systems and we think that by applying our ideas it might be possible to transform them as well. Furthermore, especially the mixing process can be applied to many other applications. Mix-nets are widely used, for instance, in electronic auctions [HGP09, BMC04] and electronic exam systems [HP10]. Furthermore, they may be of particular interest for invitations to bid and online surveys.

With respect to privacy usually only computationally bounded attackers are considered. However, this work shows that only minor changes are needed to guarantee privacy unconditionally. We hope that this becomes a common goal not only in the field of electronic voting but also for other applications where privacy of users is of special importance.

Bibliography

[AC10] Jordi Puiggalí Allepuz and Sandra Guasch Castelló. Universally verifiable efficient re-encryption mixnet. In *Electronic Voting*, pages 241–254, 2010. Cited on pages 37 and 38.

[ACG07] Roberto Araújo, Ricardo Felipe Custódio, and Jeroen van de Graaf. A verifiable voting protocol based on Farnel. In *Workshop On Trustworthy Elections (WOTE)*, June 2007. Cited on pages 7, 12, and 103.

[ACG10] Roberto Araújo, Ricardo Felipe Custódio, and Jeroen van de Graaf. A verifiable voting protocol based on Farnel. In *Towards Trustworthy Elections*, pages 274–288, 2010. Cited on page 28.

[AI03] Masayuki Abe and Hideki Imai. Flaws in some robust optimistic mixnets. In *ACISP*, pages 39–50, 2003. Cited on page 35.

[AR06] Ben Adida and Ronald L. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. In *Workshop on Privacy in Electronic Society (WPES)*, pages 29–40, 2006. Cited on pages 3 and 95.

[AR08] Roberto Araújo and Peter Y. A. Ryan. Improving the Farnel voting scheme. In *Electronic Voting*, pages 169–184, 2008. Cited on pages 12, 103, and 104.

[Ara08] Roberto Samarone dos Santos Araújo. *On Remote and Voter-Verifiable Voting*. PhD thesis, TU Darmstadt, Darmstadt, Dezember 2008. Cited on pages 105 and 112.

[BBE+13] Josh Benaloh, Michael D. Byrne, Bryce Eakin, Philip T. Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. Star-vote: A secure, transparent, auditable, and reliable voting system. In *EVT/WOTE*, 2013. Cited on page 3.

[BCD+09] Michael Backes, Tongbo Chen, Markus Dürmuth, Hendrik P. A. Lensch, and Martin Welk. Tempest in a teapot: Compromising reflections revisited. In *IEEE Symposium on Security and Privacy*, pages 315–327, 2009. Cited on page 23.

[BCH+12a] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. A supervised verifiable voting protocol for the victorian electoral commission. In *Electronic Voting*, pages 81–94, 2012. Cited on page 59.

[BCH+12b] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Vanessa Teague, Roland Wen, Zhe (Joson) Xia, and Sriramkrishnan Srinivasan. Using Prêt à Voter in Victoria State Elections. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, 2012. Cited on pages 59 and 60.

[BDG+10] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic side-channel attacks on printers. In *USENIX Security Symposium*, pages 307–322, 2010. Cited on page 23.

[BDG13] Johannes Buchmann, Denise Demirel, and Jeroen van de Graaf. Towards a publicly-verifiable mix-net providing everlasting privacy. In *Financial Cryptography*, pages 197–204, 2013. Cited on page 45.

[BDO14] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. *IACR Cryptology ePrint Archive*, 2014:75, 2014. Cited on page 101.

[Ben07] Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *Electronic Voting Technology Workshop (EVT)*, pages 14–14, 2007. Cited on page 107.

[BG02] Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *ACM Conference on Computer and Communications Security*, pages 68–77, 2002. Cited on page 35.

[BL11] Josh Benaloh and Eric Lazarus. The trash attack: An attack on verifiable voting systems and a simple mitigation. Technical Report MSR-TR-2011-115, Microsoft, 2011. Cited on page 13.

[BMC04] Mike Burmester, Emmanouil Magkos, and Vassilios Chrissikopoulos. Uncoercible e-bidding games. *Electronic Commerce Research*, 4(1-2):113–125, 2004. Cited on page 120.

[BMQR07] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo Voting: secure and coercion-free voting using a trusted random number generator. In *VOTE-ID*, pages 111–124, 2007. Cited on page 7.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC*, pages 544–553, 1994. Cited on pages 3 and 16.

[CCC+09] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE TransIFS*, 4(4):611–627, 2009. Cited on page 6.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368, 2008. Cited on page 4.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001. Cited on page 101.

[CEC+08] David Chaum, Aleksander Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan T. Sherman, and Poorvi L. Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3):40–46, 2008. Cited on page 3.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT*, pages 72–83, 1996. Cited on pages 7, 12, and 101.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, pages 103–118, 1997. Cited on page 3.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):pp. 84–88, 1981. Cited on page 19.

[Cha85] David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In *EUROCRYPT*, pages 241–244, 1985. Cited on page 27.

[Cha88] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In *EUROCRYPT*, pages 177–182, 1988. Cited on page 7.

[Cha02] David Chaum. Secret-ballot receipts and transparent integrity – improving voter confidence and electronic voting at polling places, 2002. Cited on pages 25, 37, and 38.

[Cha04] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004. Cited on page 3.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992. Cited on page 40.

[CP07] Danny De Cock and Bart Preneel. Electronic voting in Belgium: Past and future. In *VOTE-ID*, pages 76–87, 2007. Cited on page 1.

[CPP13] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In *ESORICS*, pages 481–498, 2013. Cited on page 47.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. In *ESORICS*, pages 118–139, 2005. Cited on pages 3 and 63.

[Cus01] Ricardo Custódio. Farnel: um protocolo de votação papel com verificabilidade parcial. Invited Talk at Simpósio Segurança em Informática (SSI), November 2001. Cited on page 12.

[DGA12] Denise Demirel, Jeroen van de Graaf, and Roberto Araùjo. Improving Helios with everlasting privacy towards the public. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, August 2012. Cited on pages 12 and 89.

[DH12] Denise Demirel and Maria Henning. Legal analysis of privacy weaknesses in poll-site evoting systems. *Jusletter IT - Die Zeitschrift für IT und Recht*, 2012. Cited on pages 21 and 28.

[DHG+13] Denise Demirel, Maria Henning, Jeroen van de Graaf, Peter Y. A. Ryan, and Johannes Buchmann. Prêt à voter providing everlasting privacy. In *VOTE-ID*, pages 156–175, 2013. Cited on pages 59, 74, 76, and 83.

[DHR+11] Denise Demirel, Maria Henning, Peter Y. A. Ryan, Steve Schneider, and Melanie Volkamer. Feasibility analysis of Prêt à Voter for German Federal Elections. In *VOTE-ID*, pages 158–173, 2011. Cited on pages 4, 21, 59, and 68.

[DJV12] Denise Demirel, Hugo Jonker, and Melanie Volkamer. Random block verification: Improving the Norwegian electoral mix-net. In *Electronic Voting*, pages 65–78, 2012. Cited on page 31.

[DK01] Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In *EUROCRYPT*, pages 152–165, 2001. Cited on page 18.

[ECA10] Aleksander Essex, Jeremy Clark, and Carlisle Adams. Aperio: High integrity elections for developing countries. In *Towards Trustworthy Elections*, pages 388–401, 2010. Cited on page 7.

[ECHA10] Aleksander Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. Eperio: mitigating technical complexity in cryptographic election verification. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, pages 1–16, 2010. Cited on page 8.

[FCS06] Kevin Fisher, Richard Carback, and Alan T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *Workshop on Trustworthy Elections (WOTE)*, 2006. Cited on page 71.

[Fed09] Federal Constitutional Court of Germany. Voting computer judgement. *(BVerfGE) - Judicial decisions of the Federal Constitutional Court of Germany*, 123:39, 2009. Cited on pages 1 and 31.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986. Cited on page 48.

[FS01] Pierre-Alain Fouque and Jacques Stern. Fully distributed threshold RSA under standard assumptions. In *ASIACRYPT*, pages 310–330, 2001. Cited on page 18.

[FS03] Niels Ferguson and Bruce Schneier. *Practical cryptography*. Wiley, 2003. Cited on page 27.

[GHB+06] Rop Gonggrijp, Willem-Jan Hengeveld, Andreas Bogk, Dirk Engling, Hannes Mehnert, Frank Rieger, Pascal Scheffers, and Barry Wels. Nedap/Groenendaal ES3B voting computer; a security analysis, 2006. Cited on page 23.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, pages 295–310, 1999. Cited on page 18.

[Gov] Ministry of local government and modernisation - the e-vote trial. `http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-trial.html?id=597658`. Accessed: 2014-05-23. Cited on pages 6 and 37.

[Gra09] Jeroen van de Graaf. Voting with unconditional privacy by merging Prêt à voter and PunchScan. *IEEE Transactions on Information Forensics and Security*, 4(4):674–684, 2009. Cited on page 8.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010. Cited on pages 4, 20, 47, and 62.

[GZB+02] Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *ASIACRYPT*, pages 451–465, 2002. Cited on pages 5 and 35.

[HDV12] Maria Henning, Denise Demirel, and Melanie Volkamer. Öffentlichkeit vs. Verifizierbarkeit - Inwieweit erfüllt mathematische Verifizierbarkeit den Grundsatz der Öffentlichkeit der Wahl. *Internationalen Rechtsinformatik Symposiums, IRIS 2012*, pages 213–220, 2012. Cited on page 2.

[Hea07] James Heather. Implementing STV securely in Prêt à Voter. In *CSF*, pages 157–169, 2007. Cited on page 63.

[HGP09] Jaydeep Howlader, Anushma Ghosh, and Tandra DebRoy Pal. *Secure Receipt-Free Sealed-Bid Electronic Auction*, pages 228–239. 2009. Cited on page 120.

[HL08] James Heather and David Lundin. The append-only web bulletin board. In *Formal Aspects in Security and Trust*, pages 242–256, 2008. Cited on page 32.

[HP10] Andrea Huszti and Attila Pethö. A secure electronic exam system. *Publicationes Mathematicae Debrecen*, 77/3.-4.:299–312, 2010. Cited on page 120.

[JFR13] Rui Joaquim, Paulo Ferreira, and Carlos Ribeiro. Eviv: An end-to-end verifiable internet voting system. *Computers & Security*, 32:170–191, 2013. Cited on page 3.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002. Cited on pages 20, 25, 31, and 37.

[Jon03] Douglas W. Jones. A brief illustrated history of voting. 2003. Cited on page 1.

[JRF09] Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira. Veryvote: A voter verifiable code voting system. In *VOTE-ID*, pages 106–121, 2009. Cited on page 3.

[Kal03] Burt Kaliski. RSA laboratories - TWIRL and RSA key size, May 2003. Cited on page 27.

[KMHU04] Arthur M. Keller, David Mertz, Joseph Lorenzo Hall, and Arnold B. Urken. Privacy issues in an electronic voting machine. In *WPES*, pages 33–34, 2004. Cited on page 23.

[KMO01] Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting, 2001. Cited on page 96.

[KSW05] Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, pages 33–50, August 2005. Cited on page 23.

[KTW12] Shahram Khazaei, Björn Terelius, and Douglas Wikström. Cryptanalysis of a universally verifiable efficient re-encryption mixnet. *IACR Cryptology ePrint Archive*, 2012:100, 2012. Cited on pages 6 and 38.

[Kuh04] Markus G. Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *Privacy Enhancing Technologies*, pages 88–107, 2004. Cited on page 23.

[Kuh07] Dr. Markus Kuhn. Bericht über eine Kurzuntersuchung zur Einschätzung des Risikos kompromittierender RF Abstrahlungen eines digitalen Wahlstifts, 2007. Cited on page 23.

[KW13] Shahram Khazaei and Douglas Wikström. Randomized partial checking revisited. In *CT-RSA*, pages 115–128, 2013. Cited on page 37.

[LZ12] Helger Lipmaa and Bingsheng Zhang. A more efficient computationally sound non-interactive zero-knowledge shuffle argument. In *SCN*, pages 477–502, 2012. Cited on pages 4, 20, 47, and 62.

[MN06] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *CRYPTO*, pages 373–392, 2006. Cited on pages 3, 7, and 15.

[MN07] Tal Moran and Moni Naor. Split-ballot voting: everlasting privacy with distributed trust. In *ACM Conference on Computer and Communications Security*, pages 246–255, 2007. Cited on pages 7 and 24.

[MN10] Tal Moran and Moni Naor. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Secur.*, 13(2), 2010. Cited on pages 7, 32, 47, 82, and 90.

[Nef04] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. 2004. Cited on page 3.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *EUROCRYPT*, pages 522–526, 1991. Cited on page 18.

[PH06] Stefan Popoveniuc and Ben Hosp. An introduction to punchscan. In *Workshop on Trustworthy Elections (WOTE)*, 2006. Cited on page 3.

[Pie08] Wolter Pieters. *La volonte machinale: understanding the electronic voting controversy*. UB Nijmegen [Host], 2008. Cited on page 16.

[Pie09] Wolter Pieters. Combatting electoral traces: The dutch tempest discussion and beyond. In *VOTE-ID*, pages 172–190, 2009. Cited on page 23.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT*, pages 248–259, 1993. Cited on pages 4 and 20.

[RB04] Peter Y. A. Ryan and Jeremy Bryans. A simplified version of the chaum voting scheme. Technical Report CS-TR 843, University of Newcastle upon Tyne, May 2004. Cited on page 59.

[RBH+09] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à Voter: a voter-verifiable voting system. *IEEE Trans. Inf. Forensics Security*, 4(4):662–673, 2009. Cited on pages 3, 16, 17, and 59.

[Riv06] Ronald L. Rivest. The ThreeBallot voting system, 2006. Cited on page 7.

[RP05] Peter Y. A. Ryan and Thea Peacock. Prêt à Voter: a systems perspective. Technical report, 2005. Cited on page 29.

[RP10] Peter Y. A. Ryan and Thea Peacock. A threat analysis of Prêt à Voter. In *Towards Trustworthy Elections*, pages 200–215, 2010. Cited on pages 22, 28, 59, 60, and 68.

[RS06] Peter Y. A. Ryan and Steve A. Schneider. Prêt à Voter with re-encryption mixes. In *ESORICS*, pages 313–326, 2006. Cited on page 63.

[RS07] Ronald L. Rivest and Warren D. Smith. Three voting protocols: Threeballot, VAV, and Twin. In *Electronic Voting Technology Workshop (EVT)*, pages 16–16, 2007. Cited on pages 12, 28, and 103.

[Rub81] Reuven Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1981. Cited on page 109.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991. Cited on page 40.

[Sek10] Hidenori Sekiguchi. Information leakage of input operation on touch screen monitors caused by electromagnetic noise. In *IEEE International Symposium on Electromagnetic Compatibility (EMC)*, pages 127–131, 2010. Cited on page 23.

[SFCC11] Alan T. Sherman, Russell A. Fink, Richard Carback, and David Chaum. Scantegrity III: automatic trustworthy receipts, highlighting over/under votes, and full voter verifiability. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, pages 7–23, 2011. Cited on page 3.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, pages 393–403, 1995. Cited on pages 4, 31, 35, and 36.

[Smu90] Peter Smulders. The threat of information theft by reception of electromagnetic radiation from RS-232 cables. *Computers & Security*, 9(1):53–58, 1990. Cited on page 23.

[SSC+11] Steve Schneider, Sriramkrishnan Srinivasan, Chris Culnane, James Heather, and Zhe Xia. Prêt à Voter with write-ins. In *VOTE-ID*, pages 174–189, 2011. Cited on page 63.

[Str06] Charlie E. M. Strauss. A critical review of the triple ballot voting system. part2: Cracking the triple ballot encryption. Draft Version 1.5, Verified Voting New Mexico, 2006. Cited on page 7.

[TW10] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In *AFRICACRYPT*, pages 100–113, 2010. Cited on page 4.

[VP09] Martin Vuagnoux and Sylvain Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security Symposium*, pages 1–16, 2009. Cited on page 23.

[Wik03] Douglas Wikström. Five practical attacks for "optimistic mixing for exit-polls". In *Selected Areas in Cryptography*, pages 160–175, 2003. Cited on page 35.

[XCH+10] Zhe Xia, Chris Culnane, James Heather, Hugo Jonker, Peter Y. A. Ryan, Steve A. Schneider, and Sriramkrishnan Srinivasan. Versatile Prêt à Voter: Handling multiple election methods with a unified interface. In *INDOCRYPT*, pages 98–114, 2010. Cited on pages 4, 59, and 63.

[XSH+07] Zhe Xia, Steve A. Schneider, James Heather, Peter Y. A. Ryan, David Lundin, Roger Peel, and P. Howard. Prêt à Voter: All-in-one. In

Workshop on Trustworthy Elections (WOTE), pages 47–56, 2007. Cited on page 63.

[XSHT08] Zhe Xia, Steve A. Schneider, James Heather, and Jacques Traoré. Analysis, improvement, and simplification of Prêt à Voter with Paillier encryption. In *Electronic Voting Technology Workshop (EVT)*, 2008. Cited on page 63.

Wissenschaftlicher Werdegang

November 2010 - November 2013 Wissenschaftliche Mitarbeiterin und Doktorandin bei Prof. Johannes Buchmann am Lehrstuhl für Kryptographie und Computeralgebra an der Technischen Universität Darmstadt.

Oktober 2003 - November 2010 Studium der Informatik an der Technischen Universität Darmstadt

Oktober 2002 - Oktober 2003 Ausbildung zur mathematisch- technischen Assistentin an der Georg - Kerschensteiner - Schule in Bad Homburg

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbständig verfasst habe.

Darmstadt, _____