A critique of game-based definitions of receipt-freeness for voting

Ashley Fraser[1], Elizabeth A. Quaglia[1], and Ben Smyth[2]

[1] Information Security Group, Royal Holloway, University of London, UK
{Ashley.Fraser.2016, Elizabeth.Quaglia}@rhul.ac.uk
[2] Interdisciplinary Centre for Security, Reliability and Trust,
University of Luxembourg, Luxembourg
research@bensmyth.com

Abstract. We analyse three game-based definitions of receipt-freeness; uncovering soundness issues with two of the definitions and completeness issues with all three. Hence, two of the definitions are too weak, i.e., satisfiable by voting schemes that are not intuitively receipt-free. More precisely, those schemes need not even satisfy ballot secrecy. Consequently, the definitions are satisfiable by schemes that reveal how voters' vote. Moreover, we find that each definition is limited in scope. Beyond soundness and completeness issues, we show that each definition captures a different attacker model and we examine some of those differences.

Keywords: E-voting, receipt-freeness, privacy, game-based definitions, computational security.

1 Introduction

Electronic voting, or e-voting, is the process of voting with the use of electronic aids at some stage in the voting process. In the context of this paper, we use the term e-voting to refer to remote electronic voting. That is, e-voting that does not require paper at any point in the process and can be accomplished anywhere in the world. E-voting is gaining popularity, both for public office elections and other voting scenarios. In particular, Australia has used iVote [20] for state general elections in New South Wales since 2011 and Estonia has implemented Internet voting in municipal elections since 2005 and in parliamentary elections since 2007 [38]. Moreover, Helios [2, 18] has been used by the International Association for Cryptologic Research (IACR) to elect board members since 2010 [19]. It has also been used to elect the president of Université Catholique de Louvain, Belgium, and Princeton's student government [17].

E-voting has created new opportunities, including the introduction of convenience to the voting process and the potential to automate the process of tallying elections when compared to hand-counting ballots in a traditional paper-based election. It also has the potential to produce *verifiable* elections, one of the main security goals of e-voting.[1] E-voting also creates new challenges. In particular, voter privacy is a concern. This is not new or unique to *electronic* voting but is particularly true for schemes that do not rely on a physical voting booth because the voter cannot rely upon the privacy afforded by the booth. A step towards overcoming the challenge of ensuring voter privacy is to provide rigorous privacy definitions for e-voting schemes, and then formally prove that a scheme satisfies a given definition.

Privacy for e-voting is often presented as a hierarchy of security properties [14] as follows, and there exist definitions of each property in the literature.

- *Ballot secrecy*: a voter's vote remains secret throughout the election, except when the result of the election reveals the vote, or when partial information about the vote can be deduced from the result.
- *Receipt-freeness*: a voter cannot prove their vote to anyone.
- *Coercion-resistance*: a voter can cast their vote as they intended, even if they are under the control of an attacker for some time during the election.

The relationship between these privacy properties is often considered to be linear [14]. In particular, receipt-freeness strengthens ballot secrecy with additional protection against vote buying, which ensures potential attackers have no incentive to buy votes, since a voter cannot prove how they voted, and therefore cannot prove that their vote was truly 'bought.' Coercion-resistance strengthens receipt-freeness by protecting against randomization, abstention and simulation attacks [23]. Küesters *et al.* challenge this hierarchy, showing that increasing the level of ballot secrecy can lead to a decrease in the level of coercion-resistance [25].

Formal ballot secrecy definitions were surveyed in [6, 32], where Bernhard *et al.* and Smyth compared existing ballot secrecy definitions from the literature and presented their own definitions. Similarly, definitions of coercion resistance were surveyed in [35]. Receipt-freeness, on the other hand, has not been surveyed, which motivates this work.

The earliest definitions of receipt-freeness are informal, with the first definition credited to Benaloh and Tuinstra in 1994 [5]. A general shift towards formal definitions occurred in response to concerns that voting schemes may appear to be receipt-free when they are not [31]. The early formal definitions, with the exception of Moran and Naor's simulation-based definition [30], are formulated in the symbolic model, for example, [3, 9, 15, 16, 21, 22]. These definitions use a variety of logical languages to capture the intuition of receipt-freeness. In fact, these definitions helped to shape the intuition and determine how to define receipt-freeness. However, more recently, there has been a movement towards game-based definitions of receipt-freeness, possibly driven by the simplicity of proof techniques in the model. Given that this is a young area of research and, to the best of our knowledge, there is no examination that tests the rigour of

[1] Verifiability is typically defined as individual verifiability (any voter can check that their ballot is counted), universal verifiability (anyone can check that the published tally is correct) and eligibility verifiability (only eligible voters voted). The interested reader can consult [12, 34, 37] for a discussion on the subject of verifiability.

these game-based definitions, we revisit existing game-based definitions in the literature and perform a critical analysis.

1.1 Our contributions

We analyse three game-based definitions of receipt-freeness from the literature: a receipt-freeness definition by Kiayias *et al.*, which we call KZZ [24] (§3); one by Chaidos *et al.*, which we call CCFG [10] (§4); and one by Bernhard *et al.* for schemes that use deniable vote updating, a process that allows a voter to change their vote without detection, which we call DKV [7, 8] (§5). We cast each definition into the syntax introduced in Definition 1 to facilitate analysis and comparison of definitions.

We uncover soundness issues with KZZ and CCFG, and find all three definitions to be incomplete. The soundness issue in KZZ arises because the definition is satisfied by schemes that reveal how voter's vote when not all voters vote (§3.1) and an issue arises in CCFG because Chaidos *et al.* do not consider *strong consistency* (§4.1), a property defined to accompany ballot secrecy definition BPRIV [6], upon which CCFG is based, and is used to detect some attacks against ballot secrecy. The definitions are incomplete because some schemes are out of scope. Schemes that count votes in some particular ways and others that allow voters to submit multiple ballots are out of scope of KZZ (§3.2). In particular, we prove that neither KZZ nor CCFG is satisfiable by JCJ [23] (§3.2,4.2). Finally, DKV limits the class of schemes it considers to those that use deniable vote updating.

We discuss the attacker model adopted by each definition, showing that each definition considers a different attacker model. We find that KZZ models a voter that attempts to prove their vote to an attacker, without allowing the voter to interact with the attacker before voting. In particular, the attacker cannot provide instructions to the voter (§3.3). We demonstrate that the attacker model in CCFG is much stronger, capturing an attacker with some control over the voter (§4.3). We also comment that DKV does not model a voter who attempts to prove their vote, but only asks whether an attacker can determine whether a voter has updated their vote from the attacker's choice or not. We discuss the consequences of these differing attacker models, questioning whether each definition captures the core intuition of receipt-freeness.

2 Preliminaries

Given an algorithm A, we let $A(y_1, \ldots, y_n; c)$ denote the output of A on inputs $y_1, \ldots, y_n$ and coins c, and let $A(y_1, \ldots, y_n)$ denote $A(y_1, \ldots, y_n; c)$ for some coins c chosen uniformly at random. Moreover, we let $x \leftarrow M$ denote assignment of M to x.

An e-voting scheme typically consists of the following five phases. First (*Setup*), the election administrator[2] computes and publishes public parameters of the

[2] For simplicity, we consider each entity to be a single individual but the role of any individual can be distributed.

scheme. Secondly (*Register*), the administrator provides eligible voters with a public and private credential and adds the public credential to a list $\mathcal{L}$.[3] Thirdly (*Vote*), each voter selects their vote v. This vote is stored as a ballot b on the ballot box $\mathcal{BB}$. Fourthly (*Tally*), a tallier computes and publishes the result. Finally (*Verification*), voters verify that their ballot is on the ballot box and observers verify that the tally is correct. We now formally introduce the syntax for an e-voting scheme, adapted from [6, 10], that follows this structure.

Definition 1 (E-voting scheme). *An* e-voting scheme *Γ is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Verify})$ *relative to a result function $f : \mathbb{V} \to R$ where $\mathbb{V}$ is the set of all possible votes and R is the result space such that:*

$\mathsf{Setup}(1^\lambda)$ *On input security parameter 1^λ, algorithm* Setup *outputs an election key pair pk and sk, where pk is the public key and sk is the private key.*

$\mathsf{Register}(1^\lambda)$ *On input security parameter 1^λ, algorithm* $\mathsf{Register}$ *outputs a public/private credential pair upk and usk and updates the list $\mathcal{L}$ with upk (i.e. $\mathcal{L} \leftarrow \mathcal{L} \cup \{upk\}$).*

$\mathsf{Vote}(v, usk, pk, 1^\lambda)$ *On input vote v, private credential usk, public key pk and security parameter 1^λ, algorithm* Vote *outputs a ballot b.*

$\mathsf{Append}(\mathcal{BB}, b)$ *On input ballot box $\mathcal{BB}$ and ballot b, algorithm* Append *updates $\mathcal{BB}$ to include the ballot b and outputs the updated ballot box.*

$\mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk, 1^\lambda)$ *On input ballot box $\mathcal{BB}$, list $\mathcal{L}$, private key sk and security parameter 1^λ, algorithm* Tally *computes the election outcome r, and outputs r with a tallying proof ρ that the tally is correct.*

$\mathsf{Verify}(\mathcal{BB}, r, \rho, pk, 1^\lambda)$ *On input ballot box $\mathcal{BB}$, election outcome r, proof ρ, public key pk and security parameter 1^λ, any interested party can check that the outcome of the election was computed correctly. The output of algorithm* Verify *is 1 if the election result verifies and 0 otherwise.*

E-voting schemes must satisfy correctness*: let f be a result function,*[4] *m_b be the maximum number of ballots and m_c be the maximum number of candidates. We say that Γ satisfies correctness with respect to f, m_b and m_c if there exists a negligible function* negl *such that, for all security parameters λ and choices $v_1, \ldots, v_{n_v} \in \mathbb{V}$ where n_v is an integer such that $n_v \leq m_b \wedge |\mathbb{V}| \leq m_c$,* $\Pr\big[(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda);$ *for* $i = 1, \ldots, n_v$: $\big\{(upk_i, usk_i) \leftarrow \mathsf{Register}(1^\lambda);$ $b_i \leftarrow \mathsf{Vote}(v_i, usk_i, pk, 1^\lambda);$ $\mathcal{BB} \leftarrow \mathsf{Append}(\mathcal{BB}, b_i)\big\};$ $\mathcal{L} \leftarrow \{upk_1, \ldots, upk_{n_v}\};$ $(r, \rho) \leftarrow \mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk, 1^\lambda)$: $r = f(v_1, \ldots, v_{n_v})\big] > 1 - \mathsf{negl}(\lambda)$.

Our correctness definition uses ideas from the correctness definitions in [6, 37] and considers an experiment in which the outcome is calculated in two ways: 1) the outcome is calculated in the normal way by running Tally, and 2) the outcome is computed by applying a result function f to all the votes input to Vote. Those two ways must compute equivalent outcomes to satisfy the correctness property.

[3] Not all e-voting schemes register voters, for example, Helios.

[4] The definition of correctness requires f to be correct, i.e., f must output the election outcome with respect to $v_1, \ldots, v_{n_v}$.

3 Receipt-freeness by Kiayias, Zacharias & Zhang (KZZ)

In this section, we analyse the receipt-freeness definition by Kiayias *et al.* [24], which we call KZZ. The game captures the following idea: the attacker should be unable to distinguish between a voter who submits a vote and either proves that they submitted that vote, or attempts to prove that they submitted a different vote.

Definition 2 (KZZ)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Verify})$ *be an e-voting scheme,* $\mathcal{A}$ *be an adversary,* $\mathcal{S}$ *be a simulator,*[5] λ *be a security parameter,* n_v*,* n_c *and* t *be positive integers and* β *be a bit. Let* $\mathsf{Exp}^{\mathsf{KZZ},\beta}_{\mathcal{A},\mathcal{S},\Gamma}(\lambda, n_v, n_c, t)$ *be the game that proceed as follows:*

1. *The challenger initializes* $\mathcal{BB}$ *as an empty list and inputs* $1^\lambda, n_v, n_c$ *to adversary* $\mathcal{A}$*, which outputs a set of eligible voters* $\mathcal{I} = \{id_1, \ldots, id_{n_v}\}$ *and a set of possible vote choices* $\mathbb{V}$ *such that* $|\mathbb{V}| = n_c$*.*
2. *The challenger computes* $\mathsf{Setup}(1^\lambda)$ *to produce the key pair* (pk, sk) *and, for each* $i \in \{1, \ldots, n_v\}$*, computes* $\mathsf{Register}(1^\lambda)$ *to produce a credential pair* (upk, usk)*. Public credentials are added to the list* $\mathcal{L}$*, hence,* $\mathcal{L} = \{upk_1, \ldots, upk_{n_v}\}$*. The challenger inputs* pk *and* $\mathcal{L}$ *to* $\mathcal{A}$*.*
3. *For each* $i \in \{1, \ldots, n_v\}$*,* $\mathcal{A}$ *decides whether* id_i *is corrupt.*
   – *If so, the challenger inputs* usk_i *to* $\mathcal{A}$*, which outputs a ballot* b*.*
   – *Otherwise (*id_i *is not corrupt),* $\mathcal{A}$ *outputs votes* $v_0, v_1 \in \mathbb{V}$ *to the challenger, the challenger computes ballot* $b \leftarrow \mathsf{Vote}(v_\beta, usk_i, pk, 1^\lambda)$*, and the challenger returns the ballot to* $\mathcal{A}$*, along with either the view* view *of the voter during* Vote *when* $\beta = 0$ *or* $\mathcal{S}(\mathsf{view})$ *when* $\beta = 1$*.*[6]

   Finally, the challenger computes $\mathcal{BB} \leftarrow \mathsf{Append}(\mathcal{BB}, b)$*.*
4. *The challenger computes* $(r, \rho) \leftarrow \mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk, 1^\lambda)$ *and inputs* r*,* ρ *and* $\mathcal{BB}$ *to* $\mathcal{A}$*, which outputs a bit* β'*.*
5. *The game outputs* 1 *if the following conditions are satisfied: (i)* $\beta' = \beta$*, (ii) the number of corrupted voters is bounded by* t*, and (iii)* $f(\langle v_0\rangle_{id_i \in \mathcal{V}_h}) = f(\langle v_1\rangle_{id_i \in \mathcal{V}_h})$*, i.e., with respect to uncorrupted voters, denoted by the set* $\mathcal{V}_h$*, the outcome of the election computed via the result function* f *is the same, regardless of whether* $\beta = 0$ *or* $\beta = 1$*.*

An e-voting scheme Γ *satisfies* KZZ *for* n_v *voters,* n_c *candidates and at most* t *corrupted voters if there exists a probabilistic polynomial-time simulator* $\mathcal{S}$ *and a negligible function* negl *such that, for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and all security parameters* λ*, we have*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{KZZ},0}_{\mathcal{A},\mathcal{S},\Gamma}(\lambda, n_c, n_v, t) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{KZZ},1}_{\mathcal{A},\mathcal{S},\Gamma}(\lambda, n_c, n_v, t) = 1\right] \right| \leq \mathsf{negl}(\lambda)\,.$$

We demonstrate a soundness issue with KZZ, namely, that KZZ guarantees receipt-freeness only if all voters vote (§3.1). Moreover, KZZ is incomplete because there exists schemes that are receipt-free but do no satisfy KZZ (§3.2).

[5] Simulator $\mathcal{S}$ models a voter providing fake evidence of a vote they did not submit.

[6] *view* is defined as the "internal state of the voter" [24]. It refers to any information that the voter inputs to the voting client to produce a ballot, including, but not necessarily limited to, private credentials and the coins input to algorithm Vote.

3.1 Soundness issue

KZZ requires that a single ballot is submitted to the ballot box on behalf of each voter. As a result, KZZ declares schemes as receipt-free that reveal how voters vote, when not all voters vote. To illustrate this, consider an e-voting scheme for at most n_v voters. If less than n_v voters vote and, hence, $|\mathcal{BB}| \leq n-1$, define algorithm Tally to output an election outcome $r = \{(id_1, v_1), \ldots\ (id_i, v_i)\}$ where $i \leq n_v - 1$, i.e., it lists each voter that voted and the vote submitted by that voter. Clearly, this scheme is not receipt-free. Indeed, the scheme does not satisfy ballot-secrecy because the result announces the link between voter and vote. However, in the KZZ game, a ballot must be submitted for every voter, so this privacy leakage will not be identified. Therefore, the scheme may satisfy KZZ whilst not being receipt-free. Consequently, a proven secure scheme may leak every voter's vote when a real-world deployment cannot ensure that all voter's vote. Hence, there may exist schemes that are proven secure but, in practice, do not offer any degree of privacy for voters.

3.2 Completeness issues

Schemes with multiple ballots are out of scope: KZZ requires the submission of a single ballot on behalf of each voter. Yet, some e-voting schemes require the submission of more than one ballot to achieve receipt-freeness. For instance, e-voting schemes may use fake private credentials (that are indistinguishable from real private credentials). Such schemes require voters to cast dummy ballots using fake credentials and prove the contents of dummy ballots (rather than real ballots) to an attacker. A voter can then cast a ballot for a different vote using their real credential. In these schemes it is necessary that a voter submits two ballots in order to submit a vote but prove that they submitted a different vote. JCJ [23] is an e-voting scheme that achieves receipt-freeness this way, hence, the scheme cannot satisfy KZZ.

Proposition 1. JCJ *does not satisfy* KZZ.

A proof of Proposition 1 appears in Appendix A.

KZZ limits the set of result functions for which a scheme can be declared receipt-free: We demonstrate this limitation, which exists as a consequence of the condition $f(\langle v_0 \rangle_{id_i \in \mathcal{V}_h}) = f(\langle v_1 \rangle_{id_i \in \mathcal{V}_h})$, by considering an informal argument used by Bernhard *et al.* in [6] to show that ballot-secrecy definition PRIV [4] has the same limitation. Consider an e-voting scheme with two possible candidate choices, namely $\mathbb{V} = \{0, 1\}$, for which f outputs the winning candidate, or '0' in the event of a draw. An adversary against the KZZ game can submit a ballot for '1' on behalf of a corrupted voter and can submit votes on behalf of all other voters such that $\langle v_0 \rangle_{id_i \in \mathcal{V}_h}$ has exactly half entries equal to '0' and half equal to '1', and $\langle v_1 \rangle_{id_i \in \mathcal{V}_h}$ has all entries equal to '0'. Then, $f(\langle v_0 \rangle_{id_i \in \mathcal{V}_h}) = f(\langle v_1 \rangle_{id_i \in \mathcal{V}_h}) = 0$, but the election outcome $r = 0$ (if $\beta = 0$) or 1 (if $\beta = 1$). Thus, the adversary can output $\beta' = \beta$ and the scheme does not satisfy KZZ.

3.3 Further discussion

KZZ models attack scenarios in which a voter provides evidence of their vote (including their private credential) to the attacker only *after* voting, thereby assuming that honest voters do not reveal their private credentials until they have voted. We illustrate that DEMOS, an e-voting scheme that satisfies KZZ [24, Theorem 5], is no longer receipt-free if an attacker can compel a voter to reveal their credentials before voting, that is, when the assumption does not hold.

DEMOS provides each eligible voter with a voting card (which is a private credential in our terminology). This voting card consists of two parts: the first part contains a list of candidates and a unique vote code associated with each candidate. This is repeated on the second part of the voting card, although the vote codes associated with each candidate are different. To cast a ballot, each voter selects a part of their voting card (part '0' or part '1', which we call the coins, using our terminology) and inputs the selected part and the vote code listed next to their chosen candidate to the voting client. The part of the ballot and the vote code constitute the voter's ballot. The ballot box is updated with the ballot, i.e., algorithm Append outputs $\mathcal{BB} \parallel b$. Intuitively, DEMOS satisfies KZZ because voters can swap vote codes on the voting card, and can make the vote code on their ballot correspond to any candidate they wish. Therefore, the voter can convince the attacker that the submitted vote code corresponds to the attacker's choice of candidate.

However, consider the following scenario: an attacker wants a voter to vote for candidate A but the voter wants to vote for candidate B. The attacker requests to see the voter's voting card *before* voting. Only after seeing the voting card, the attacker requests that the voter cast a ballot for candidate A. In this scenario, the voter may not have switched vote codes for candidates A and B. Thus, the voter cannot vote for candidate A *and* convince the attacker that they voted for candidate B. In contrast, if an attacker does not see the voting card until after voting, the voter can switch the vote codes for candidates A and B. Therefore, DEMOS provides a guarantee of receipt-freeness only if the voting card is revealed after voting.

That being said, KZZ does appear to capture the correct intuition of receipt-freeness. As noted in §1, receipt-freeness aims to prevent a voter from proving their vote and KZZ captures the scenario in which the voter votes and then attempts to prove their vote. The scenario above describes an attacker who interacts with a voter before voting, which is outside the scope of KZZ. The question is: should this attack scenario be captured by receipt-freeness, or does it fall under the remit of coercion-resistance? We do not address this in our informal definition of receipt-freeness (§1) because this is a grey area in the literature. For instance, Delaune *et al.* define receipt-freeness as the property that "a voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way" and coercion-resistance as "a voter cannot cooperate with a coercer to prove to him that she voted in a certain way" [15]. This suggests that providing information to an attacker before voting is captured by coercion-resistance, not receipt-freeness. In fact, Delaune *et al.*'s definition of receipt-freeness implies that a voter uses information to prove their vote *after* voting, whereas providing information to an attacker *before* voting is considered cooperation with an attacker. It appears that KZZ captures this intuition. On the other hand, some authors take a different approach. We will discuss a different approach that leads to an entirely different conclusion in §4. For now, we note that establishing a boundary between receipt-freeness and coercion-resistance is an open problem.

4 Receipt-freeness by Chaidos *et al.* (CCFG)

In this section, we consider a definition of receipt-freeness by Chaidos *et al.* [10], which we call CCFG. Chaidos *et al.* consider ballot boxes that contain ballots validated by an algorithm Valid and consider ballot boxes as private, introducing an algorithm Publish that outputs a public view of a ballot box, which we call the bulletin board. Formally, Chaidos *et al.* extend the definition of an e-voting scheme (Definition 1) to include algorithms Valid and Publish such that:

Valid$(\mathcal{BB}, b)$ On input ballot box $\mathcal{BB}$ and a ballot b, algorithm Valid outputs $\top$, if the ballot is valid, or $\perp$ otherwise.

Publish$(\mathcal{BB})$ On input ballot box $\mathcal{BB}$, algorithm Publish outputs bulletin board $\mathcal{PBB}$.

Furthermore, algorithm Verify is redefined to take as input a bulletin board $\mathcal{PBB}$, *rather than* a ballot box $\mathcal{BB}$. All other aspects of Verify remain the same.[7]

In this context, Chaidos *et al.* define CCFG as an extension of the ballot secrecy game BPRIV by Bernhard *et al.* [6]. CCFG captures the idea that the attacker should be unable to determine whether, throughout the game, they are viewing a real or fake election, when the outcome is always computed for the real election. As such, CCFG models two ballot boxes, $\mathcal{BB}_0$ and $\mathcal{BB}_1$, and, respectively, two bulletin boards, $\mathcal{PBB}_0$ and $\mathcal{PBB}_1$. The adversary must determine whether they are viewing $\mathcal{PBB}_0$ or $\mathcal{PBB}_1$, when the outcome is always computed over the contents of $\mathcal{BB}_0$.

CCFG relies on algorithms SimSetup and SimProof, which facilitate the ability to simulate the tallying proof ρ such that the outcome computed over the contents of $\mathcal{BB}_0$ appears to be computed over the contents of $\mathcal{BB}_1$, when $\beta = 1$. Algorithms SimSetup and SimProof are defined as follows:

SimSetup(1^λ) On input security parameter 1^λ, algorithm SimSetup outputs an election key pair pk and sk and auxiliary information aux, which is used to output a simulated proof during the tally phase of the election.

SimProof$(\mathcal{BB}, r, aux)$ On input ballot box $\mathcal{BB}$, election outcome r and auxiliary information aux, algorithm SimProof outputs a proof ρ that r is the outcome of an election computed over the contents of $\mathcal{BB}$.

[7] In this section, we use the term e-voting scheme to refer to Definition 1 plus algorithms Valid and Publish, unless stated otherwise.

Using those algorithms, CCFG is formalized as follows:

Definition 3 (CCFG). *Let Γ = ($\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Publish}, \mathsf{Verify}$) be an e-voting scheme, $\mathcal{A}$ be an adversary, λ be a security parameter and β be a bit. Let $\mathsf{Exp}^{\mathsf{CCFG},\beta}_{\mathcal{A},\Gamma}(\lambda)$ be the game that proceeds as follows:[8] the challenger initializes $\mathcal{BB}_0$ and $\mathcal{BB}_1$ as empty lists and V_r and V_c as empty sets. Adversary $\mathcal{A}$ can then query the oracles defined in Figure 1, under the constraint that $\mathcal{O}\mathsf{setup}$ must be queried before any other oracles and $\mathcal{O}\mathsf{tally}$ appears only as the final oracle call. The adversary terminates by outputting a bit β'. The game outputs 1 if $\beta' = \beta$.*

An e-voting scheme Γ satisfies CCFG if there exists algorithms $\mathsf{SimSetup}$ and $\mathsf{SimProof}$ and a negligible function negl such that, for all probabilistic polynomial-time adversaries $\mathcal{A}$ and all security parameters λ, we have

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{CCFG},0}_{\mathcal{A},\Gamma}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{CCFG},1}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)\,.$$

We show that CCFG is unsound as it overlooks the needs for strong consistency (§4.1) and is incomplete, limiting the class of schemes that can be declared receipt-free (§4.2).

4.1 Soundness issue

A property called *strong consistency* is introduced in [6] to accompany BPRIV. Strong consistency requires that the outcome output by Tally is consistent with the application of result function f to the votes and is necessary to detect tally policies that may lead to an attack against ballot secrecy. Therefore, as noted in [6, Section IV.D], an e-voting scheme must satisfy BPRIV *and* strong consistency to achieve ballot secrecy. However, Chaidos *et al.* do not consider this property in [10], which results in an unsound definition of receipt-freeness. In fact, there exists schemes satisfying CCFG that are vulnerable to attacks that violate ballot secrecy. We briefly recall an example in [6, Section IV.D], that illustrates this: define an e-voting scheme for two candidates (say, A and B) that outputs a multiset of the submitted votes as the election outcome. Suppose this scheme satisfies CCFG. Now, define a modified scheme such that, if the first voter votes for candidate A, this vote is removed from the election outcome. An adversary against CCFG cannot distinguish games $\mathsf{Exp}^{\mathsf{CCFG},0}_{\mathcal{A},\Gamma}(\lambda)$ and $\mathsf{Exp}^{\mathsf{CCFG},1}_{\mathcal{A},\Gamma}(\lambda)$, where Γ is the modified scheme, because the tally is always computed over the contents of $\mathcal{BB}_0$ and so the election outcome will be the same in both games. However, through removal of the first vote, the tally for this modified scheme allows the adversary to determine whether the first vote is for candidate A or B. Therefore, the modified scheme reveals how the first voter voted. We refer the reader to [6, Section IV.D] for a full details of this argument. Unfortunately, CCFG cannot simply adopt the original definition of strong consistency by Bernhard *et al.*, because it is defined over different syntax, in particular, the original definition does not consider consider algorithm Append. Adapting the original definition to consider this algorithm is a possible direction for future work.

[8] We omit $\mathsf{SimSetup}$ and $\mathsf{SimProof}$ as inputs to game $\mathsf{Exp}^{\mathsf{CCFG},\beta}_{\mathcal{A},\Gamma}(\lambda)$ for simplicity.

4.2 Completeness issue

We observe that CCFG is unsatisfiable by schemes for which $\mathsf{Append}(\mathcal{BB}, b)$ outputs $\mathcal{BB} \parallel b$ and $\mathsf{Publish}(\mathcal{BB})$ outputs $\mathcal{BB}$. That is, $\mathsf{Append}(\mathcal{BB}, b)$ appends ballot b to ballot box $\mathcal{BB}$ without processing the ballot in any way and $\mathsf{Publish}(\mathcal{BB})$ outputs $\mathcal{BB}$ such that the ballot that appears on the public view of $\mathcal{BB}$ is identical to the ballot submitted by the voter. Formally, we have the following result.

Proposition 2. *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Publish}, \mathsf{Verify})$ *be an e-voting scheme for which* $\mathsf{Append}(\mathcal{BB}, b)$ *outputs* $\mathcal{BB} \parallel b$ *and* $\mathsf{Publish}(\mathcal{BB})$ *outputs* $\mathcal{BB}$*. Then* Γ *does not satisfy* CCFG.

Proof. We construct an adversary $\mathcal{A}$ against the CCFG game as follows. $\mathcal{A}$ queries $pk \leftarrow \mathcal{O}\mathsf{setup}()$, $upk \leftarrow \mathcal{O}\mathsf{register}(id)$ and $(upk, usk) \leftarrow \mathcal{O}\mathsf{corrupt}(id)$. Then, $\mathcal{A}$ computes $b_0 \leftarrow \mathsf{Vote}(v_0, usk, pk, 1^\lambda)$ and $b_1 \leftarrow \mathsf{Vote}(v_1, usk, pk, 1^\lambda)$ and queries $\mathcal{O}\mathsf{receipt}(id, b_0, b_1)$, $\mathcal{PBB}_\beta \leftarrow \mathcal{O}\mathsf{board}()$ and $(r, \rho) \leftarrow \mathcal{O}\mathsf{tally}()$. It follows that $\mathcal{PBB}_\beta$ contains the single entry b_0 (if $\beta = 0$) or b_1 (if $\beta = 1$). Therefore, $\mathcal{A}$ can correctly distinguish $\mathsf{Exp}^{\mathsf{CCFG},0}_{\mathcal{A},\Gamma}(\lambda)$ and $\mathsf{Exp}^{\mathsf{CCFG},1}_{\mathcal{A},\Gamma}(\lambda)$ and outputs $\beta' = \beta$. Thus, the e-voting scheme Γ does not satisfy CCFG. □

CCFG is unsatisfiable by these schemes because, in the CCFG game, the adversary submits two ballots to $\mathcal{O}\mathsf{receipt}$. To satisfy CCFG, the adversary must be unable to distinguish a bulletin board that contains ballot b_0 and a bulletin board that contains ballot b_1, where the adversary queries $\mathcal{O}\mathsf{receipt}(id, b_0, b_1)$ in the CCFG game. This requires that ballots are modified in some way before they are appended to $\mathcal{BB}_0$ and $\mathcal{BB}_1$, or before $\mathcal{PBB}_\beta$ is published. Otherwise, the adversary can trivially distinguish as shown in the proof of Proposition 2. Partly, CCFG excludes these schemes by design. Chaidos *et al.* acknowledge that a scheme satisfies CCFG only if it achieves receipt-freeness without the voter relying on some evasion strategy [10]. Generally, schemes that provide voters with an evasion strategy, a procedure that the scheme provides to allow the voter to evade coercion, do not rely on ballot modification but instead on the use of an evasion strategy to achieve receipt-freeness. This means that schemes that rely on evasion strategies to achieve receipt-freeness cannot satisfy CCFG despite the fact that they are receipt-free. For example, JCJ relies on fake credentials, a type of evasion strategy, to achieve receipt-freeness (§3.2). Thus, we have the following corollary.

Corollary 1. JCJ *does not satisfy* CCFG.

The corollary follows from Proposition 2, since JCJ ballots are not modified before they are appended to the ballot box and $\mathsf{Publish}(\mathcal{BB})$ outputs $\mathcal{BB}$.

4.3 Further discussion

CCFG captures the scenario in which an honest voter constructs their ballot and gives the attacker the coins used (or possibly uses coins provided by the attacker) to construct their ballot. This allows the attacker to reconstruct the ballot locally and then check whether the ballot appears on the bulletin board. CCFG captures this scenario through the oracle $\mathcal{O}\mathsf{receipt}$, which allows the adversary to construct ballots on behalf of voters and then submit these ballots to $\mathcal{O}\mathsf{receipt}$. The adversary can then view $\mathcal{PBB}_\beta$, and expects to see a ballot corresponding to one of those submitted to $\mathcal{O}\mathsf{receipt}$.

Chaidos *et al.* take a very different approach to the intuition of receipt-freeness than Kiayias *et al.* As mentioned in §3.3, Delaune *et al.* consider a voter that cooperates with an attacker (e.g. by using coins provided by the attacker) to fall outwith the scope of receipt-freeness. Moreover, Kiayias *et al.* exclude this scenario from the definition of KZZ. However, Chaidos *et al.* consider this to fall within the scope of receipt-freeness although, admittedly, they do refer to CCFG as a definition of *strong* receipt-freeness. Therefore, we see that there is no consensus over the boundary between receipt-freeness and coercion-resistance in the literature and that definitions of receipt-freeness capture varying intuitions.

5 Receipt-freeness for deniable vote updating by Bernhard, Kulyk & Volkamer (DKV)

In this section, we analyse a definition of receipt-freeness by Bernhard *et al.* [8, 7] for schemes that use *deniable vote updating*, which we call DKV. Bernhard *et al.* construct a game-based definition of receipt-freeness for KTV-Helios [26], a variant of the Helios e-voting scheme that uses deniable vote updating whereby a voter casts a ballot, and then changes their vote, without an attacker detecting the change. Deniable vote updating uses re-voting to allow the voter to change their vote and an algorithm executed by an election administrator that posts dummy ballots on behalf of voters in order to hide the fact that a voter may cast more than one ballot for the purposes of deniable vote updating. This is the definition of deniable vote updating for the KTV-Helios voting scheme as defined in [8], but variations exist [1, 28, 29]. In [8, Section 4.1] it was recognized that CCFG does not apply to KTV-Helios because deniable vote updating is a type of evasion strategy and the strategy is required to achieve receipt-freeness. Therefore, Bernhard *et al.* introduce a new receipt-freeness definition that modifies CCFG to schemes that use deniable vote updating. We rely on the definition presented in [7] (the technical report associated with the conference version of the paper [8]).

Bernhard *et al.* capture the following idea: the attacker should be unable to distinguish a voter who submits a vote and a voter who submits the same vote but then deniably updates their vote, where the adversarial advantage of distinguishing is denoted δ. DKV adopts e-voting syntax (Definition 1) extended with algorithm Valid (§4) and considers timestamps such that algorithm Vote is redefined to take additional input of a timestamp t, indicating the time at which a ballot is to be cast. DKV relies on algorithms $\mathsf{SimSetup}$ and $\mathsf{SimProof}$ (§4) and, additionally, requires algorithms $\mathsf{DenyUpdate}$ and $\mathsf{Obfuscate}$ such that:

$\mathsf{DenyUpdate}(v_0, v_1, usk, t_u, pk, 1^\lambda)$ On input votes v_0, v_1, private credential usk, timestamp t_u chosen uniformly at random from some probability distribution

$\mathbb{P}$, public key pk and security parameter 1^λ, algorithm $\mathsf{DenyUpdate}$ outputs a ballot that updates a vote from vote v_0 to vote v_1 at timestamp t_u.

$\mathsf{Obfuscate}(\mathcal{BB}, id)$ On input ballot box $\mathcal{BB}$ and voter id, algorithm $\mathsf{Obfuscate}$ casts dummy ballots for voter id to hide ballots cast by id in the event that id deniably updates their vote, and outputs the updated ballot box.

Using those algorithms, DKV is formalized as follows:

Definition 4 (DKV)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Verify})$ *be an e-voting scheme with timestamps,* $\mathcal{A}$ *be an adversary,* λ *be a security parameter and* β *be a bit. Let* $\mathsf{Exp}^{\mathsf{DKV},\beta}_{\mathcal{A},\Gamma}(\lambda)$ *be the game that proceeds as follows: the challenger initializes* $\mathcal{BB}_0$ *and* $\mathcal{BB}_1$ *as empty lists. If* $\beta = 0$ *(resp.,* $\beta = 1$*), the challenger computes* $\mathsf{Setup}(1^\lambda)$ *to produce the keypair* (pk, sk) *(resp., computes* $\mathsf{SimSetup}(1^\lambda)$ *to produce the keypair* (pk, sk) *and auxiliary information aux) and, for each* $i \in \{1, \ldots, n_v\}$*, computes* $\mathsf{Register}(1^\lambda)$ *to produce a credential pair* (upk, usk)*. Public credentials are added to the list* $\mathcal{L}$*, namely,* $\mathcal{L} = \{usk_1, \ldots, usk_{n_v}\}$*. The challenger inputs pk,* $\mathcal{L}$ *and* $\mathcal{BB}_\beta$[9] *to adversary* $\mathcal{A}$*. Adversary* $\mathcal{A}$ *can then query the oracles defined in Figure 2, under the constraint that* $\mathcal{O}\mathsf{receipt}$ *can be queried at most once and* $\mathcal{O}\mathsf{tally}$ *appears only as the final oracle call. The adversary terminates by outputting a bit* β'*. The game outputs* 1 *if* $\beta' = \beta$*.*

An e-voting scheme Γ *satisfies* DKV *if there exists algorithms* $\mathsf{DenyUpdate}$*,* $\mathsf{Obfuscate}$*,* $\mathsf{SimSetup}$ *and* $\mathsf{SimProof}$ *and a negligible function* negl *such that, for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and all security parameters* λ*, we have*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{DKV},0}_{\mathcal{A},\Gamma}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{DKV},1}_{\mathcal{A},\Gamma}(\lambda) = 1\right] - \delta \right| \leq \mathsf{negl}(\lambda)\,.$$

We did not find any soundness issues with DKV. In particular, although DKV uses the same framework as CCFG, DKV does not overlook the need for strong consistency and defines strong consistency in their syntax in the technical report [7]. Clearly, DKV is incomplete because it limits the class of e-voting schemes that can be declared receipt-free to schemes with timestamps that achieve receipt-freeness through the use of deniable vote updating, though this is by design.

Again, Bernhard *et al.* capture a different intuition of receipt-freeness. DKV does not model a voter who interacts with an attacker to prove their vote. In other words, DKV does not model a voter that provides an attacker with any proof of their vote. In particular, there is no mechanism to capture the fact that a voter may try to pass their credentials or coins to an attacker. Certainly, this definition does not pose any issues with respect to whether it captures attack scenarios

[9] In this game $\mathcal{BB} = \mathcal{PBB}$. Bernhard *et al.* do not mention adversarial access to $\mathcal{BB}_\beta$ in the technical report [7] but do allow the adversary to 'see' $\mathcal{BB}$ in the conference version [8]. We assume that, as DKV is a modification of CCFG, the adversary should have access to $\mathcal{BB}_\beta$. This could be resolved by providing the adversary with access to an oracle $\mathcal{O}\mathsf{publish}$ as defined for CCFG. This provides the adversary with a view of $\mathcal{BB}_\beta$, which we assume is the intention in this definition.

that should be considered under the heading of coercion-resistance. However, it does raise questions about whether this definition captures receipt-freeness. As there is no mechanism for a voter to attempt to prove their vote, we conclude that receipt-freeness is guaranteed under the assumption that the voter does not pass any proof of their vote to the attacker.

6 Conclusion

In this paper, we systematically analysed game-based definitions of receipt-freeness and uncovered completeness and soundness issues. We also found that each definition considers a different attacker model. In this section we bring together these findings and suggest possible directions for future research.

We discovered soundness issues with KZZ and CCFG: we proved that KZZ can be satisfied by schemes that leak every voter's vote. Moreover, we found that CCFG does not consider strong consistency, which seems necessary for soundness. By comparison, DKV considers strong consistency, and we believe coupling CCFG with a suitable notion of strong consistency should suffice to achieve soundness, albeit defining such a notion is non-trivial.

We found each definitions to be incomplete. KZZ requires that each voter votes, and only once. As a result, JCJ does not satisfy KZZ. CCFG is unsatisfiable by a class of schemes that do not process ballots before adding them to the ballot box and for which the bulletin board is identical to the ballot box. Consequently, JCJ does not satisfy CCFG either. Furthermore, DKV only applies to schemes that use deniable vote updating. Thus, there is no game-based definition of receipt-freeness that can be applied to a wide class of schemes. This is a potential area of future research.

Beyond completeness and soundness issues, each definition captures a different attacker model: KZZ models a voter that provides evidence of their vote (including coins and credentials) *after* voting. By comparison, CCFG captures scenarios wherein the voter uses coins provided by an attacker. Consequently, KZZ does not capture scenarios where a voter interacts with an attacker before voting (e.g., by providing the attacker with credentials), whereas CCFG does. It is unclear whether a definition of receipt-freeness should capture the scenario where a voter interacts with an attacker before voting, or whether this should be considered beyond the scope of receipt-freeness and be captured by coercion-resistance. The boundary between receipt-freeness and coercion-resistance is unclear and we believe establishing a boundary is an interesting open problem.

We also observe that KZZ, CCFG and DKV consider that *all* election authorities are honest, in particular, the election administrator, tallier and ballot box are honest. Beyond trusting the authorities, communication channels between voters and/or election authorities are considered to be private. In practice, trust assumptions may be difficult to enforce, or it may not be possible to prove that the assumption holds. Motivated by this, ballot secrecy in the context of a malicious ballot box was considered in [33, 32], whereby the adversary controls the contents of the ballot box. We believe that this setting warrants further exploration and we believe that security definitions with minimal trust assumptions are preferable.

A further point of interest is that receipt-freeness (and, more generally, privacy for e-voting) does not exist in a vacuum and it must be considered in the context of other desirable security properties. This has been addressed in the recent literature and one notable area of research relates to the relationship between receipt-freeness and verifiability. Traditionally, verifiability and privacy are viewed as competing and, in some cases, incompatible properties. For example, if a voter wants to keep their vote secret but also wants to be sure that their vote is included in the tally, the scheme must provide the voter with enough information to verify the result, without compromising the privacy of their vote. Some results have shown that the relationship between privacy and verifiability is rather intricate: Chevallier-Mames *et al.* prove that receipt-freeness and universal verifiability are incompatible under certain assumptions on the communication channels and election authorities [11]. On the other hand, receipt-freeness and universal verifiability are compatible under different assumptions, see, for example [27, 10]. Moreover, Cortier and Lallemand recently showed that ballot secrecy implies individual verifiability [13] assuming the same trust assumptions for both ballot secrecy and individual verifiability, but this result does not hold more generally [36]. We believe that exploring the relationship between privacy and verifiability, particularly with respect to trust assumptions, is an interesting and exciting area of future research.

Acknowledgements. This work is partly supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/P009301/1), and by the Luxembourg National Research Fund (FNR) under the FNR-INTER-VoteVerif project (10415467).

References

1. Dirk Achenbach, Carmen Kempka, Bernhard Löwe, and Jörn Müller-Quade. Improved coercion-resistant electronic elections through deniable re-voting. *USENIX Journal of Election Technology and Systems*, pages 26–45, 2015.
2. Ben Adida. Helios: web-based open-audit voting. In *USENIX Security Symposium*, volume 17, pages 335–348. USENIX, 2008.
3. Anguraj Baskar, Ramaswamy Ramanujam, and SP Suresh. Knowledge-based modelling of voting protocols. In *11th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71. ACM, 2007.
4. Josh Benaloh. *Verifiable secret-ballot elections.* PhD thesis, Yale Univeristy, 2006.
5. Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *26th Annual ACM Symposium on Theory of Computing*, pages 544–553. ACM, 1994.
6. David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: a comprehensive analysis of game-based ballot privacy definitions. In *IEEE Security and Privacy*, pages 499–516. IEEE, 2015.
7. David Bernhard, Oksana Kulyk, and Melanie Volkamer. Security proofs for participation privacy, receipt-freeness, ballot privacy, and verifiability against malicious

bulletin board for the helios voting scheme. Cryptology ePrint Archive, Report 2016/431.

8. David Bernhard, Oksana Kulyk, and Melanie Volkamer. Security proofs for participation privacy, receipt-freeness and ballot privacy for the helios voting scheme. In *12th International Conference on Availability, Reliability and Security*, page 1. ACM, 2017.
9. Katharina Braunlich and Rudiger Grimm. Formalization of receipt-freeness in the context of electronic voting. In *6th International Conference on Availability, Reliability and Security*, pages 119–126. IEEE, 2011.
10. Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: a non-interactive receipt-free electronic voting scheme. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1614–1625. ACM, 2016.
11. Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. *Towards Trustworthy Elections*, 6000:191–199, 2010.
12. Véronique Cortier, David Galindo, Ralf Küsters, Johannes Mueller, and Tomasz Truderung. Sok: verifiability notions for e-voting protocols. In *IEEE Security and Privacy*, pages 779–798. IEEE, 2016.
13. Véronique Cortier and Joseph Lallemand. Voting: you can't have privacy without individual verifiability. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 53–66. ACM, 2018.
14. Stephanie Delaune, Steve Kremer, and Mark Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *19th Computer Security Foundations Workshop*, pages 28–42. IEEE, 2006.
15. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
16. Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. A formal taxonomy of privacy in voting protocols. In *International Conference on Communications (ICC)*, pages 6710–6715. IEEE, 2012.
17. Harvard magazine secret ballots, verifiable votes. `harvardmagazine.com/2010/05/secret-ballots-verifiable-votes`. Accessed: 2017/08/01.
18. Helios voting system. `heliosvoting.org/`. Accessed: 2018/03/06.
19. IACR final report of iacr electronic voting committee. `www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html`. Accessed: 2017/08/01.
20. iVote online voting. `www.ivote.nsw.gov.au/`. Accessed: 2017/08/01.
21. HL Jonker and W Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *IAVoSS Workshop On Trustworthy Elections (WOTE)*, 2006.
22. Hugo L Jonker and Erik P de Vink. Formalising receipt-freeness. In *International Conference on Information Security*, pages 476–488. Springer, 2006.
23. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *ACM Workshop on Privacy in the Electronic Society*, pages 61–70. ACM, 2005.
24. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 468–498. Springer, 2015.
25. Ralf Küesters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: new insights from a case study. In *IEEE Security and Privacy*, pages 538–553. IEEE, 2011.

26. Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. Extending helios towards private eligibility verifiability. In *International Conference on E-Voting and Identity*, pages 57–73. Springer, 2015.
27. Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. *International Conference on Information Security and Cryptology*, pages 245–258, 2004.
28. Philipp Locher and Rolf Haenni. Receipt-free remote electronic elections with everlasting privacy. *Annals of Telecommunications*, 71(7-8):323–336, 2016.
29. Philipp Locher, Rolf Haenni, and Reto E Koenig. Coercion-resistant internet voting with everlasting privacy. In *International Conference on Financial Cryptography and Data Security*, pages 161–175. Springer, 2016.
30. Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Annual International Cryptology Conference*, pages 373–392. Springer, 2006.
31. Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *International Workshop on Security Protocols*, pages 25–35. Springer, 1998.
32. Ben Smyth. Ballot secrecy: security definition, sufficient conditions, and analysis of helios. Cryptology ePrint Archive, Report 2015/942.
33. Ben Smyth. Ballot secrecy with malicious bulletin boards. Cryptology ePrint Archive, Report 2014/822.
34. Ben Smyth. A foundation for secret, verifiable elections. Cryptology ePrint Archive, Report 2018/225.
35. Ben Smyth. Surveying definitions of coercion resistance. Cryptology ePrint Archive, Report 2019/822.
36. Ben Smyth. Verifiability of helios mixnet. Cryptology ePrint Archive, Report 2018/017.
37. Ben Smyth, Steven Frink, and Michael R. Clarkson. Election verifiability: cryptographic definitions and an analysis of helios, helios-c, and jcj. Cryptology ePrint Archive, Report 2015/233.
38. Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J Alex Halderman. Security analysis of the estonian internet voting system. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 703–715. ACM, 2014.

A JCJ does not satisfy KZZ and CCFG

We briefly define the JCJ voting scheme in the syntax of Definition 1, omitting full details of the non-interactive zero-knowledge (NIZK) proofs, Tally and Verify algorithms. We refer the reader to [37, Appendix I] for a full scheme description.

Definition 5. *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA, multipicatively homomorphic asymmetric encryption scheme in a given group $\mathbb{G}$, $\mathbf{\Sigma}$ be a set of sigma protocols and $\mathcal{H}$ be a hash function used to form NIZK proof systems. We define $\mathsf{JCJ}(\Pi, \mathbf{\Sigma}, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Verify})$ as follows:*

$\mathsf{Setup}(1^\lambda)$ *Compute $(pk_\Pi, sk_\Pi) \leftarrow \mathsf{Gen}(1^\lambda)$, a NIZK proof of correct key construction pf and output $pk = (pk_\Pi, pf)$ and $sk = (pk_\Pi, sk_\Pi))$.*

$\mathsf{Register}(1^\lambda)$ *Compute $usk \leftarrow_\$ \mathbb{G}$; $upk \leftarrow \mathsf{Enc}(pk_\Pi, usk)$. Output (upk, usk).*

$\mathsf{Vote}(v, usk, pk, 1^\lambda)$ *Compute* $\mathsf{ciph}_1 \leftarrow \mathsf{Enc}(pk_\Pi, v)$; $\mathsf{ciph}_2 \leftarrow \mathsf{Enc}(pk_\Pi, usk)$, *a NIZK proof of plaintext knowledge of* ciph_1*, denoted* σ*, and a NIZK proof of conjunctive plaintext knowledge of* ciph_1 *and* ciph_2*, denoted* τ*. Output ballot* $b = (\mathsf{ciph}_1, \mathsf{ciph}_2, \sigma, \tau)$.

$\mathsf{Append}(\mathcal{BB}, b)$ *Output* $\mathcal{BB} \leftarrow \mathcal{BB} \parallel b$.

$\mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk, 1^\lambda)$ *Remove invalid and duplicate ballots and mix the remaining ballots. Remove ineligible ballots and decrypt remaining ballots. Output* (r, ρ) *where* r *is a tally vector and* ρ *is a NIZK proof that each step of* Tally *is carried out correctly.*

$\mathsf{Verify}(\mathcal{BB}, r, \rho, pk, 1^\lambda)$ *Verify that each step of* Tally *is carried out correctly. Output 1 if each step verifies.*

We now formally state and prove Proposition 1.

Proposition 3. *Let* JCJ' *be the scheme defined in Definition 5 for* n_v *voters,* n_c *candidates and at most* t *corrupted voters. Then* JCJ' *does not satisfy* KZZ*.*

Proof. Let $view = (v, usk, c)$ where v is the vote submitted, usk is the private credential of the voter and c is the coins used to encrypt vote v and private credential usk. We construct an adversary $\mathcal{A}$ against the KZZ game as follows.

On input 1^λ, n_v, n_c and t, $\mathcal{A}$ outputs a set of possible votes $\mathcal{I}$ and a set of candidates $\mathbb{V}$ such that $|\mathcal{I}| = n_v$ and $|\mathbb{V}| = n_c$. Then, on input $\mathcal{L}$ and pk, $\mathcal{A}$ chooses voter id_1 to be uncorrupted and submits (v_0, v_1), where v_0 and v_1 are distinct choices, on behalf of id_1 and the challenger inputs a ballot b and $view_1$ to $\mathcal{A}$. The adversary continues in this manner, selecting that each of the $n-1$ remaining voters are uncorrupted and submitting pairs of votes of the form (v_0, v_1) on behalf of each voter such that $f(\langle v_0 \rangle_{id_i \in \mathcal{V}_h}) = f(\langle v_1 \rangle_{id_i \in \mathcal{V}_h})$. Finally, on input (r, ρ) and $\mathcal{BB}$, $\mathcal{A}$ outputs a bit β'.

Consider voter id_1. $\mathcal{BB}$ contains the ballot $b = \mathsf{Vote}(v_0, usk_1, pk, 1^\lambda; c_0)$ (if $\beta = 0$) or $b = \mathsf{Vote}(v_1, usk_1, pk, 1^\lambda; c_1)$ (if $\beta = 1$) where usk_1 is the private credential of voter id_1 and c_0, c_1 are the coins used to encrypt votes v_0 and v_1, respectively. $\mathcal{A}$ can locally compute a ballot b' using $view_1$. If $\beta = 0$, $view_1 = (v_0, usk, c_0)$ and $b' = b$. If $\beta = 1$, $view_1 = (v_0, usk', c')$ for some simulated private credential usk' and simulated coins c', and $b' \neq b$ (assuming JCJ satisfies injectivity, the property that algorithm Vote does not map two distinct votes to the same ballot, which is shown to be true in [37]). Therefore, $\mathcal{A}$ can correctly distinguish $\mathsf{Exp}^{\mathsf{KZZ},0}_{\mathcal{A},\mathcal{S},\mathsf{JCJ}'}(\lambda, n_v, n_c, t)$ and $\mathsf{Exp}^{\mathsf{KZZ},1}_{\mathcal{A},\mathcal{S},\mathsf{JCJ}'}(\lambda, n_v, n_c, t)$ and outputs $\beta' = \beta$. Furthermore, $\mathcal{A}$ does not corrupt any voters (hence, the number of corrupted voters is less than t) and $f(\langle v_0 \rangle_{id_i \in \mathcal{V}_h}) = f(\langle v_1 \rangle_{id_i \in \mathcal{V}_h})$. Thus, the e-voting scheme JCJ' does not satisfy KZZ. □

We now define the JCJ e-voting scheme in the syntax of Chaidos *et al.* (§4).

Definition 6. *We define* $\mathsf{JCJ}''(\Pi, \mathbf{\Sigma}, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Verify}, \mathsf{Publish})$ *to be the e-voting scheme* JCJ *defined in Definition 5 with the addition of algorithms* Valid *and* $\mathsf{Publish}$ *such that:*

$\mathsf{Valid}(\mathcal{BB}, b)$ *On input ballot box* $\mathcal{BB}$ *and a ballot* b*, algorithm Valid outputs* $\top$*.*

$\mathsf{Publish}(\mathcal{BB})$ *On input ballot box* $\mathcal{BB}$*, algorithm* $\mathsf{Publish}$ *outputs* $\mathcal{BB}$*.*

Then we have the formal statement of Corollary 1.

Corollary 2. JCJ'' *does not satisfy* CCFG*.*

The corollary follows from Proposition 2.

```
$\mathcal{O}\mathsf{setup}()$
if $\beta = 0$ then
    $(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda)$
else
    $(pk, sk, aux) \leftarrow \mathsf{SimSetup}(1^\lambda)$
return $pk$

$\mathcal{O}\mathsf{vote}(id, v_0, v_1)$
if $v_0, v_1 \in \mathbb{V} \wedge (id, upk, usk) \in \mathsf{V}_r$ then
    $b_0 \leftarrow \mathsf{Vote}(v_0, usk, pk, 1^\lambda)$
    $b_1 \leftarrow \mathsf{Vote}(v_1, usk, pk, 1^\lambda)$
    $\mathcal{BB}_0 \leftarrow \mathsf{Append}(\mathcal{BB}_0, b_0)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b_1)$

$\mathcal{O}\mathsf{register}(id)$
if $(id, upk, usk) \notin \mathsf{V}_r$ then
    $(upk, usk) \leftarrow \mathsf{Register}(1^\lambda)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{upk\}$
    $\mathsf{V}_r \leftarrow \mathsf{V}_r \cup \{(id, upk, usk)\}$
return $upk$

$\mathcal{O}\mathsf{cast}(id, b)$
if $\mathsf{Valid}(\mathcal{BB}_\beta, b) = \top$ then
    $\mathcal{BB}_0 \leftarrow \mathsf{Append}(\mathcal{BB}_0, b)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b)$

$\mathcal{O}\mathsf{corrupt}(id)$
if $(id, upk, usk) \in \mathsf{V}_r$ then
    $\mathsf{V}_c \leftarrow \mathsf{V}_c \cup \{(id, upk)\}$
return $(upk, usk)$

$\mathcal{O}\mathsf{tally}()$
if $\beta = 0$ then
    $(r, \rho) \leftarrow \mathsf{Tally}(\mathcal{BB}_0, \mathcal{L}, sk, 1^\lambda)$
else
    $(r, \rho') \leftarrow \mathsf{Tally}(\mathcal{BB}_0, \mathcal{L}, sk, 1^\lambda)$
    $\rho \leftarrow \mathsf{SimProof}(\mathcal{BB}_1, r, aux)$
return $(r, \rho)$

$\mathcal{O}\mathsf{board}()$
return $\mathsf{Publish}(\mathcal{BB}_\beta)$

$\mathcal{O}\mathsf{receipt}(id, b_0, b_1)$
if $(id, upk) \in \mathsf{V}_c \wedge \mathsf{Valid}(\mathcal{BB}_0, b_0) = \top \wedge \mathsf{Valid}(\mathcal{BB}_1, b_1) = \top$ then
    $\mathcal{BB}_0 \leftarrow \mathsf{Append}(\mathcal{BB}_0, b_0)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b_1)$
```

Fig. 1. Oracles used in the receipt-freeness game CCFG by Chaidos *et al.* [10]

```
$\mathcal{O}\mathsf{vote}(id, v_0, v_1, t)$
$b_0 \leftarrow \mathsf{Vote}(v_0, usk, t, pk, 1^\lambda)$
$b_1 \leftarrow \mathsf{Vote}(v_1, usk, t, pk, 1^\lambda)$
**if** $\mathsf{Valid}(\mathcal{BB}_\beta, b_\beta) = \top$ **then**
    $\mathcal{BB}_0 \leftarrow \mathsf{Append}(\mathcal{BB}_0, b_0)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b_1)$

$\mathcal{O}\mathsf{cast}(id, b)$
**if** $\mathsf{Valid}(\mathcal{BB}_\beta, b)$ **then**
    $\mathcal{BB}_0 \leftarrow \mathsf{Append}(\mathcal{BB}_0, b)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b)$

$\mathcal{O}\mathsf{tally}()$
**if** $\beta = 0$
    $(r, \rho) \leftarrow \mathsf{Tally}(\mathcal{BB}_0, \mathcal{L}, sk, 1^\lambda)$
**else**
    $(r, \rho') \leftarrow \mathsf{Tally}(\mathcal{BB}_0, \mathcal{L}, sk, 1^\lambda)$
    $\rho \leftarrow \mathsf{SimProof}(\mathcal{BB}_1, r, aux)$
**return** $(r, \rho)$

$\mathcal{O}\mathsf{receipt}(id, v_0, v_1, t)$
**if** $v_0, v_1 \in \mathbb{V}$ **then**
    $b_0 \leftarrow \mathsf{Vote}(v_0, usk, t, pk, 1^\lambda)$
    $\mathcal{BB}_0 \leftarrow \mathsf{Append}(\mathcal{BB}_0, b_0)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b_0)$
    $t_u \leftarrow_\$ \mathbb{P}$
    $b_1 \leftarrow \mathsf{DeniablyUpdate}(v_0, v_1, usk, t_u, pk, 1^\lambda)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Append}(\mathcal{BB}_1, b_1)$
    $\mathcal{BB}_0 \leftarrow \mathsf{Obfuscate}(\mathcal{BB}_0, id)$
    $\mathcal{BB}_1 \leftarrow \mathsf{Obfuscate}(\mathcal{BB}_1, id)$
```

Fig. 2. Oracles used in the receipt-freeness game DKV by Bernhard *et al.* [7]