On the Security Properties of e-Voting Bulletin Boards

Aggelos Kiayias[1,3], Annabell Kuldmaa[2], Helger Lipmaa[2,4], Janno Siim[2,5](✉), and Thomas Zacharias[1]

[1] University of Edinburgh, Edinburgh, UK
{akiayias,tzachari}@inf.ed.ac.uk
[2] University of Tartu, Tartu, Estonia
annabell.kuldmaa@gmail.com, helger.lipmaa@gmail.com, janno.siim@gmail.com
[3] IOHK, Edinburgh, UK
[4] Cybernetica-Smartmatic CEIV, Tartu, Estonia
[5] STACC, Tartu, Estonia

Abstract. In state-of-the-art e-voting systems, a bulletin board (BB) is a critical component for preserving election integrity and availability. We introduce a framework for the formal security analysis of the BB functionality modeled as a distributed system. Our framework treats a secure BB as a robust public transaction ledger, defined by Garay *et al.* [Eurocrypt 2015], that additionally supports the generation of receipts for successful posting. Namely, in our model, a secure BB system achieves *Persistence* and *Liveness* that can be *confirmable*, in the sense that any malicious behavior can be detected via a verification mechanism.

As a case study for our framework, we analyze security guarantees and weaknesses of the BB system of [CSF 2014]. We demonstrate an attack revealing that the said system does not achieve Confirmable Liveness in our framework, even against covert adversaries. In addition, we show that special care should be taken for the choice of the underlying cryptographic primitives, so that the claimed fault tolerance threshold of $N/3$ out-of N corrupted IC peers is preserved.

Next, based on our analysis, we introduce a new BB protocol that upgrades the [CSF 2014] protocol. We prove that it tolerates any number less than $N/3$ out-of N corrupted IC peers both for Persistence and Confirmable Liveness, against a computationally bounded general Byzantine adversary. Furthermore, Persistence can also be Confirmable, if we distribute the AB (originally a centralized entity in [CSF 2014]) as a replicated service with honest majority.

Keywords: Bulletin board · E-voting · Liveness · Persistence

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No. 653497 (project PANORAMIX) and No. 780477 (project PRIViLEDGE). Lipmaa and Siim were also supported by the Estonian Research Council grant (PRG49). Siim has been supported by European Regional Development Fund under the grant no. EU48684.

D. Catalano and R. De Prisco (Eds.): SCN 2018, LNCS 11035, pp. 505–523, 2018.
https://doi.org/10.1007/978-3-319-98113-0_27

1 Introduction

An electronic voting (e-voting) system is a salient instance of a network protocol where verifying the correctness of the execution is of critical importance. One can argue that if the concerned parties can not agree on the election transcript, then the voting process itself is meaningless. Besides e-voting, verifiability of the execution is desired in applications such as auctions and blockchain transactions.

It becomes apparent that in any protocol where consensus on the outcome is essential, the protocol infrastructure must guarantee a *consistent view* to all involved parties as far as auditing is concerned. Consistency here informally suggests that any two auditors engaging in the verification process on the same input but from possibly different network locations, should agree on their verdict, i.e. they both accept or reject the execution outcome. If this guarantee cannot be provided, then an adversary controlling the network could easily partition the parties into small "islands", such that each island has access to a partial, and possibly (partially) fake, view of the execution. By doing this, the adversary can undermine the auditors' consensus on the outcome.

Consistency in voting may be realized in various ways depending on the election setting. In small-scale elections (e.g. board elections) a consistent view can be achieved by executing a consensus protocol by the voters themselves, even without encrypting the votes if privacy is not a concern. However, when considering the large scale setting (e.g., national elections) where complete connectivity among the participants is unrealistic, a publicly accessible and reliable method is required for posting and reading all necessary election information. This is provided by an electronic *bulletin board (BB)* which, abstractly, encompasses two types of operations: (1) a *posting* operation involving *users* who make post requests for items of their choice, potentially subject to some access policy, and a subsystem of *item collectors (ICs)* that receive and store the submitted items. (2) A *publishing* operation, where the IC subsystem publishes the stored items on an *audit board (AB)* from where any party can read. The IC and the AB could be distributed or centralized, or even managed by the same entity. Nonetheless, the above description typifies the way BB's are treated in the e-voting literature.

It is of high importance that the BB functionality implemented by the IC and AB should function as an *immutable database*, so that submitted items cannot be erased or changed. The desired features of such a database include: (a) the ability to authenticate item contributors, (b) distributed operations to protect against attacks on availability, (c) a predetermined time-span where item submission is enabled, (d) resilience to modification so as to facilitate verifiability.

The necessity of a consistent BB has been stressed many times in e-voting literature. In his PhD thesis, Benaloh [3] assumes BBs with append-only write operations for auditing, also stressing that "implementing such bulletin boards may be a problem unto itself." Subsequently, most verifiable e-voting systems proposed (e.g. [1,4,5,7–9,11,17,23,25]) refer to the notion of BB as a fundamental component of their infrastructure without explicitly realizing it.

Despite the widely accepted critical importance of building reliable BBs for e-voting, the literature on proposals of secure and efficient BB constructions is scarce. Outside a limited number of early works [15,21,26,28,29], the most concrete examples include the BB applied in the vVote e-voting system [14] (cf. [6,12]) and the BB of the D-DEMOS Internet-voting (i-voting) system [10]. In all these cases, the introduced BB was either an integral part of a specific e-voting system [10,15], or, even though modular, lacked formal treatment under a comprehensive security model [14,21,26,28,29].

In this work, we focus on the functionality of the BB as used in e-voting systems, yet we note that our approach can be extended to other applications where a public reliable auditing system is needed. We aim to establish a complete formal treatment of a BB and propose an efficient and provably secure construction that can be deployed in a wide class of e-voting designs.

Initially, we are motivated by the security requirements proposed by Culnane and Schneider [14], suggesting that a secure BB should prevent data injection and removal, while allowing only the publishing of non-clashing items. On top of these properties, [14] prescribes a liveness guarantee of the eventual publishing of all submitted items for which got a *receipt* for correct recording. Taking a step further, we introduce a framework for the formal study of the BB concept and its security. Our framework is inspired by the notion of a *robust public transaction ledger* (RPTL) defined by Garay *et al.* [18] and the security model presented by Chondros *et al.* [10], thereby utilizing the connection between blockchain and BB systems, which, albeit being folklore, was never formalized. We define a secure BB system in a way that it can be seen as an RPTL that additionally supports the generation of receipts for successful posting. Expanding the security model for blockchain protocols of [18], we divide security into properties named *Persistence* and *Confirmable Liveness*. Confirmability in liveness captures the receipt generation capability. Persistence can also be *Confirmable*, meaning that dishonest AB behavior is detectable via verification of published data.

Next, we apply our framework for the security analysis of the BB system of [14], which we refer to as the CS BB, that utilizes standard signature and threshold signature schemes (TSS) as cryptographic building blocks. In the threat model of [14], an adversary may corrupt less than $N_c/3$ out-of the total N_c IC peers, hence we also assume this fault-tolerance threshold.

We find that CS is not secure in our framework for the $< N_c/3$ threshold. Specifically, we demonstrate an attack showing that CS with N_c IC peers does not achieve Confirmable Liveness. Our attack falls outside the threat model of [14] but raises a discussion about its plausibility. In particular, the threat model of [14] relies on a "fear of detection" (cf. the full version of [14], [13, Sect. 8]), to exclude certain adversarial protocol deviations in the IC subsystem. Nevertheless, such covert security reasoning (cf. [2]) is not formalized or implemented in [13] and as our attack demonstrates, the detection of protocol deviation is impossible by IC peers themselves given their local protocol view.

A second, though less crucial, finding for the security of CS concerns its Confirmable Persistence. Namely, we show that for CS to achieve Confirmable Persistence under the $<N_c/3$ fault tolerance threshold, the underlying TSS should not be applied as 'black-box' and care should be taken for the choice of the

TSS construction. We briefly describe the issue in Sect. 4.3, but leave a more thorough treatment for the full version of this paper [24].

Based on our analysis, we modify CS by designing a new Publishing protocol that achieves consensus among the honest IC peers on the posted items that should be published. Combined with the CS Posting protocol, we obtain a new BB system that achieves Persistence and Confirmable Liveness for $<N_c/3$ corrupted IC peers. Persistence can also be Confirmable, if we distribute the AB, such that data posting is done by broadcasting to all AB peers and data reading is done by honest majority. The new BB system is secure against (i) any computationally bounded Byzantine adversary, (ii) in a partially synchronous setting (cf. [16]), where the message delivery delay and the synchronization loss among the entities' clocks are bounded, and the bounds themselves can be unknown within a given wide range of protocol parameters.

Summary of Contributions. Our contributions are as follows:

- The first complete framework for the study of e-voting BBs captured by the properties of Confirmable Liveness and (Confirmable) Persistence.
- Analysis of the CS BB system [14] in our security framework that reveals two vulnerabilities. In particular, one of the vulnerabilities challenges the reasoning of liveness in the threat model provided in [13, Sect. 8].
- A modified variant of the CS protocol that restores Confirmable Liveness. We prove security in our framework with an $N_c/3$ threshold for the IC subsystem against computationally bounded Byzantine adversaries. In particular, (i) Persistence holds in the asynchronous model and can be also Confirmable given honest majority of AB peers, while (ii) Confirmable Liveness holds in the partially synchronous model.

Related Work. In a wide range of state-of-the-art e-voting systems, such as [1,5,8,9,23,25], the BB is a centralized single point of failure for security analysis. Dini [15] proposed a distributed e-voting service based on [17], focusing on the service in general rather on the BB system. Several works on distributed e-voting BB solutions lacked formal security analysis, providing only constructions without proof [21,26,28,30], a study of requirements [20] or being applicable only to the kiosk-voting based setting [4]. D-DEMOS [10] is a distributed internet-voting system which adopts [25] to the distributed setting. The security in [10] is studied in a model that is a stepping stone for our framework, yet security argumentation targets specifically the D-DEMOS requirements. The CS BB system [14] is a reference point for our work, and will be analyzed in Sect. 4.

2 Preliminaries

We use κ as the security parameter. We write $f(\kappa) = \mathsf{negl}(\kappa)$ if function f is negligible in κ. We denote $[N] := \{1, 2, \ldots, N\}$ for any $N \in \mathbb{N}$.

Signature Schemes. A *signature scheme* $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ consists of: (i) the *key generation algorithm* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$ that generates a signing key sk and a verification key pk; (ii) the *signing algorithm* Sig that for message m returns $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(m)$; (iii) the *verification algorithm* $\mathsf{Vf}_{\mathsf{pk}}(m, \sigma)$ that returns 0 or 1. DS is correct if $\mathsf{Vf}_{\mathsf{pk}}(m, \mathsf{Sig}_{\mathsf{sk}}(m)) = 1$. The security of DS is formalized via the notion of *existential unforgeability against chosen message attacks* (EUFCMA).

Threshold Signature Schemes. Let $t_s < N$ be two positive integers and $P_1, \ldots, P_N$ be a set of peers. A (non-interactive) *threshold signature scheme (TSS)* $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{Combine}, \mathsf{TVf})$ consists of: (i) the *distributed key generation algorithm* $\mathsf{DistKeygen}(1^\kappa, t_s, N)$ that generates a keypair $(\mathsf{tsk}_i, \mathsf{pk}_i)$ for each peer P_i and a public key pk; (ii) the *signing algorithm* $\mathsf{ShareSig}_{\mathsf{tsk}_i}(m)$ that returns a signature share σ_i of the message m; (iii) the *share verification algorithm* $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, m, (i, \sigma_i))$ that returns 0 or 1; (iv) the *share combining algorithm* $\mathsf{Combine}(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, m, (i, \sigma_i)_{i \in S})$ that if $|S| \geq t_s + 1$, outputs a full signature $\sigma \leftarrow \mathsf{TSign}(\mathsf{tsk}, m)$ on m; (v) the *verification algorithm* $\mathsf{TVf}_{\mathsf{pk}}(m, \sigma)$ that returns 0 or 1.

The correctness of TSS requires that for $(\mathsf{tsk}, \mathsf{pk}, \mathsf{tsk}_1, \ldots, \mathsf{tsk}_N, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ output by $\mathsf{DistKeygen}(1^\kappa, t_s, N)$, if $\mathcal{S} \subseteq [N]$ s.t. $|\mathcal{S}| = t_s + 1$, it holds that (i) $\sigma_i = \mathsf{ShareSig}_{\mathsf{tsk}_i}(m)$, and (ii) if $\sigma = \mathsf{Combine}(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, m, (i, \sigma_i)_{i \in \mathcal{S}})$, then $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, m, (i, \sigma_i)) = 1$ for $i \in \mathcal{S}$ and $\mathsf{TVf}_{\mathsf{pk}}(m, \sigma) = 1$.

TSS is (t_s, N)-*EUFCMA-secure* if every PPT adversary $\mathcal{A}$ has $\mathsf{negl}(\kappa)$ advantage in performing a successful EUFCMA forgery for a message m^*, even when the sum of (i) the number of the parties $\mathcal{A}$ corrupts, and (ii) the number of parties for which $\mathcal{A}$ made a signing query for m^*, is no more than t_s.

TSS is said to be (t_s, N)-*robust*, if $\mathcal{A}$ controlling t_s peers, can not prevent honest peers from creating a valid signature. Robustness can only be achieved for $t_s < N/2$ (cf., Gennaro *et al.* [19]).

3 Framework

We introduce a formal framework for secure e-voting BB systems. First, we provide an abstract description of the consisting entities and protocols. Then, building upon the requirements stated in [14] and the modeling approach of [10, 18], we formalize BB security via the notions of *(Confirmable) Persistence* and *Confirmable Liveness*.

3.1 Syntax of a Bulletin Board System

Entities. A BB system involves the following entities: (1) a *setup authority* SA that generates the setup information and initializes all other entities with their private inputs; (2) the *users* that submit post requests for items of their choice. An item can be any data the user intends to be published, e.g., the voters' ballots, the election results or any necessary audit information; (3) a subsystem of *item collection (IC) peers* $P_1, \ldots, P_{N_c}$ that are responsible for (i) interacting with the users for posting all submitted items, and (ii) interacting with the AB (see below) to publish the recorded items; (4) a subsystem of *audit board (AB) peers* $AB_1, \ldots, AB_{N_w}$ where all the posted items are published.

Setup. During setup, SA specifies a *posting policy* $\mathcal{P} = (\mathsf{Accept}, \mathsf{Select}(\cdot))$, where

(1) $\mathsf{Accept} = \{(U, x)\}$ is a binary relation over pairs of user IDs and items. For a user U that wants to post item x, $(U, x) \in \mathsf{Accept}$ is a check the IC peers execute to initiate interaction with U for posting x. E.g., a user that is authenticated as a voter may be accepted to post a vote, but nothing else.
(2) $\mathsf{Select}(\cdot)$ is a *selection function* over sets of items defined as follows: let X_U be the set of published items associated with posts from user U. Then, $\mathsf{Select}(X_U) \subseteq X_U$ contains all valid published items posted by U, resolving any conflict among clashing items. E.g., in Estonian e-voting [22], only voter's last vote must count. Thus, if the votes were submitted in time ascending order as $x_1, x_2, \ldots, x_m$, then we set $X_U = \{x_1, x_2, \ldots, x_m\}$ and $\mathsf{Select}(X_U) = \{x_m\}$.

The SA initializes other entities with the description of $\mathcal{P}$. Next, all entities engage in a setup interaction such that when finalized, each entity has a private input (e.g., a signing key or an authentication password) and some public parameters params.

BB Protocols. The BB functionality comprises the **Posting** and **Publishing** protocols, accompanied by two verification algorithms: (i) $\mathsf{VerifyRec}$, run by the users to verify the successful posting of their items, and (ii) $\mathsf{VerifyPub}$, run by any party for auditing the validity of the data on the AB.

The **Posting** protocol is initiated by a user U that on private input s_U submits a post request for item x. Namely, U uses s_U to generate a credential cr_U[1]. Then, the user and the IC peers engage in an interaction that results in U obtaining a receipt $\mathsf{rec}[x]$ for the successful posting of x. Upon receiving $\mathsf{rec}[x]$, and using public election parameters params, U may run the algorithm $\mathsf{VerifyRec}$ on input $(\mathsf{rec}[x], x, s_U, \mathsf{params})$, that either accepts or rejects.

In the **Publishing** protocol, the IC peers upload their local records of posted items to the AB subsystem. The protocol may encompass a consensus protocol among the AB peers to agree whether a local record is admissible. In addition, any auditor may run $\mathsf{VerifyPub}$ on input params and (a subset of the) published data to check consistency of AB.

3.2 Introducing Our Security Framework

Culnane and Schneider [14] propose 4 properties that a secure BB must satisfy:

[1] E.g., if s_U is a signing key, then cr_U could be a valid signature under s_U; if s_U is a password, then cr_U can be the pair (U, s_U).

(bb.1). *Only items that have been posted may appear on the AB.* This property expresses safety against illegitimate data injection.
(bb.2). *Any item that has a valid receipt must appear on the AB.*
(bb.3). *No clashing items must both appear on the AB.*
(bb.4). *Once published, no items can be removed from the AB.* According to this property, the AB subsystem is an *append-only* posting site.

In this section, we integrate the above 4 properties into a security framework. At a high level, our framework conflates the formal approach in distributed e-voting security of Chondros *et al.* [10] with the notion of a *robust public transaction ledger (RPTL)* proposed by Garay *et al.* [18]. Namely, we view a secure BB as an RPTL that additionally provides *receipts of successful posting* for honestly submitted items. The security properties of an RPTL stated in [18] are informally expressed as follows:

- *Persistence*: once an honest peer reports an item x as posted, then all honest peers either (i) agree on the position of x on AB, or (i) not report x.
- θ-*Liveness*: honest peers report honestly submitted items in a delay bound θ.

Persistence and Liveness in the e-Voting Scenario. In the e-voting setting, honest users should get a valid receipt when engaging at the **Posting** protocol (within some time θ) that confirms the eventual publishing of the respective item. An important observation is that this property that we call θ-*Confirmable Liveness* and (bb.3) can not be satisfied concurrently if we assume that honest users may submit post requests for clashing items (e.g., multiple voting in Estonia [22]). To resolve this conflict, we do not require that (bb.3) holds and the subset of valid published items is specified via the selection function $\mathsf{Select}(\cdot)$. Given the above, we require that *Persistence* encompasses (bb.1) and (bb.4), and conflict resolution is achieved by applying $\mathsf{Select}(\cdot)$ on the AB view. Furthermore, we extend Persistence by taking into account an AB subsystem that is fully controlled by the adversary. This is formalized by the *Confirmable Persistence* property, where we require that any malicious AB behavior will be detected via the $\mathsf{VerifyPub}$ algorithm.

System Clocks. Like in [10], we assume that there exists a *global clock* variable $\mathsf{Clock} \in \mathbb{N}$, and that every system entity X is equipped with an *internal clock* variable $\mathsf{Clock}[X] \in \mathbb{N}$. We define the following two events:

- $\underline{\text{The event } \mathsf{Init}(X)}$: $\mathsf{Clock}[X] \leftarrow \mathsf{Clock}$, that initializes X by synchronizing its internal clock with the global clock.
- $\underline{\text{The event } \mathsf{Inc}(\mathsf{Clock}[X])}$: $\mathsf{Clock}[X] \leftarrow \mathsf{Clock}[X] + 1$, that causes a clock $\mathsf{Clock}[X]$ to advance by one time unit.

Synchronicity and Message Delay. We parameterize our threat model by (i) an upper bound δ on the delay of message delivery, and (ii) an upper bound Δ on the synchronization loss of the nodes' internal clocks w.r.t. the global clock. By convention, we set $\Delta = \infty$ to denote the fully *asynchronous* setting and $\delta = \infty$, to denote that the adversary may drop messages. Values $\delta, \Delta \in [0, \infty)$ refer to *partially synchronous* model, if δ, Δ are unknown.

Notation. We denote by N_c, N_w the number of IC and AB peers, respectively, and by n (an upper bound) on the number of users. In our security analysis, the parameters N_c, N_w, n are assumed polynomial in security parameter κ. Let $\mathbf{E} := \{\mathsf{SA}\} \cup \{U_\ell\}_{\ell\in[n]} \cup \{P_i\}_{i\in[N_c]} \cup \{AB_j\}_{j\in[N_w]}$ be the set of all involved BB system entities. We denote by t_c (resp. t_w) the number of IC (resp. AB) peers that the adversary may statically corrupt out of the total N_c (resp. N_w) peers. We denote the local record of IC peer P_i at global time $\mathsf{Clock} = T$ as the set of accepted and confirmed items $L_{\mathsf{post},i,T} := \{x_1, \ldots, x_{K_{i,T}}\}$, where $K_{i,T} \in \mathbb{N}$. Similarly, the AB view of peer AB_j at global time $\mathsf{Clock} = T$ is denoted as the set of items $L_{\mathsf{pub},j,T} := \{x_1, \ldots, x_{M_{j,T}}\}$, where $M_{j,T} \in \mathbb{N}$.

3.3 (Confirmable) Persistence Definition

We define Persistence via a security game $\mathcal{G}_{\mathsf{Prst}}^{\mathcal{A},\delta,\Delta,t_c,t_w}(1^\kappa, \mathbf{E})$ between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The game is also parameterized by the eventual message delivery and synchronization loss upper bounds δ and Δ. The adversary $\mathcal{A}$ may statically corrupt up to t_c out-of the N_c total IC peers and t_w out-of the N_w total AB peers, and may also choose to corrupt users. $\mathcal{C}$ initializes the BB system on behalf of the SA. Then, $\mathcal{C}$ and $\mathcal{A}$ engage in the **Setup** phase and the **Posting** and **Publishing** protocols, where $\mathcal{C}$ acts on behalf of the honest entities. Intuitively, the goal of $\mathcal{A}$ is to successfully attack the (bb.1) property (condition (P.1) in Fig. 1) or the (bb.4) property (condition (P.2) in Fig. 1).

We extend the Persistence notion by defining Confirmable Persistence. Now, the entire AB may be malicious and deviate from the **Publishing** protocol, yet the adversary fails if its attack is detected via the $\mathsf{VerifyPub}$ algorithm, on the input view of any AB peer. Formally, Confirmable Persistence is defined via the game $\mathcal{G}_{\mathsf{C.Prst}}^{\mathcal{A},\delta,\Delta,t_c}(1^\kappa, \mathbf{E})$ that follows the same steps as $\mathcal{G}_{\mathsf{Prst}}^{\mathcal{A},\delta,\Delta,t_c,t_w}(1^\kappa, \mathbf{E})$, for the special case $t_w = N_w$, except the following differences in the winning conditions for $\mathcal{A}$: (i) for every $k \in [N_w]$, the published data on AB_k should always verify successfully, and (ii) the inconsistent AB_j referred in the winning conditions may be any (malicious) AB peer. Detailed description of both games is given in Fig. 1. We define Persistence and Confirmable Persistence as follows.

Definition 1 ((Confirmable) Persistence). *Let κ be the security parameter, $N_c, N_w, t_c, t_w \in \mathbb{N}$, $\delta, \Delta \in [0, +\infty]$, and* **BB** *be a BB system with N_c IC peers and N_w AB peers. We say that* **BB** achieves Persistence *for fault-tolerance thresholds (t_c, t_w), delay message bound δ and synchronization loss bound Δ, if for every PPT adversary $\mathcal{A}$ it holds that $\mathbf{Pr}\big[\mathcal{G}_{\mathsf{Prst}}^{\mathcal{A},\delta,\Delta,t_c,t_w}(1^\kappa, \mathbf{E}) = 1\big] = \mathsf{negl}(\kappa)$.*

We say that **BB** achieves Confirmable Persistence *for fault tolerance threshold t_c, delay message bound δ and synchronization loss bound Δ, if for every PPT adversary $\mathcal{A}$, it holds that $\mathbf{Pr}\big[\mathcal{G}_{\mathsf{C.Prst}}^{\mathcal{A},\delta,\Delta,t_c}(1^\kappa, \mathbf{E}) = 1\big] = \mathsf{negl}(\kappa)$.*

Threat model.

(I). The adversary $\mathcal{A}$ statically corrupts up to t_c (resp. t_w) out-of the N_c (resp. N_w) total peers of the IC (resp. AB) subsystem. Then, $\mathcal{A}$ provides $\mathcal{C}$ with the set $L_{\mathsf{corr}} \subset \mathbf{E}$ of corrupted parties. Throughout the game, $\mathcal{C}$ plays the role of honest entities that include SA.

(II). When an honest entity X wants to transmit a message $\mathbf{M}$ to an honest entity Y, then it just sends $(X, \mathbf{M}, Y)$ to $\mathcal{A}$. If the honest entity X sends $(X, \mathbf{M}, Y)$ to $\mathcal{A}$, when the global time is $\mathsf{Clock} = T$, then $\mathcal{A}$ must write M on the incoming network tape of Y by the time that $\mathsf{Clock} = T + \delta$ (*eventual message delivery*).

(III). $\mathcal{A}$ may write on the incoming network tape of any honest entity.

(IV). $\mathcal{A}$ may invoke the event $\mathsf{Inc}(\mathsf{Clock}[X])$ under the restriction that for any X, $|\mathsf{Clock}[X] - \mathsf{Clock}| \leq \Delta$ (*loose clock synchronization*).

Protocol execution under the presence of $\mathcal{A}$.

- The challenger initiates the **Setup** phase playing the role of SA and determines the posting policy $\mathcal{P} = \big(\mathsf{Accept}, \mathsf{Select}(\cdot)\big)$. Then, it initializes every system entity $X \in \mathbf{E}$ by running the event $\mathsf{Init}(X)$.
- Upon initialization, $\mathcal{C}$ and $\mathcal{A}$ engage in the **Setup** phase and the **Posting** and **Publishing** protocols, where $\mathcal{C}$ acts on behalf of the honest entities.
- For each user U_ℓ, $\ell \in [n]$, $\mathcal{A}$ may choose to corrupt, and thus fully control U_ℓ.
- The adversary $\mathcal{A}$ may provide $\mathcal{C}$ with a message $(\mathsf{post}, U_\ell, x)$ for some honest user U_ℓ and an item x of its choice. Upon receiving $(\mathsf{post}, U_\ell, x)$, $\mathcal{C}$ engages in the **Posting** protocol on behalf of U_ℓ. Upon successful interaction, $\mathcal{C}$ obtains a receipt $\mathsf{rec}[x]$ for x.

Winning conditions for $\mathcal{G}_{\mathsf{Prst}}^{\mathcal{A},\delta,\Delta,t_c,t_w}$.

The game outputs 1 iff there is an AB peer $AB_j \notin L_{\mathsf{corr}}$, at least one of the following holds:

(P.1). There are t_c+1 honest IC peers $\{P_{i_k}\}_{k\in[t_c+1]}$, an item x, and moments T, T', such that (i) $T \leq T'$, (ii) $x \in L_{\mathsf{pub},j,T}$, and (iii) $x \notin L_{\mathsf{post},i_k,T'}$, for any $k \in [t_c+1]$.

(P.2). There is an item x and moments T, T' such that (i) $T < T'$, (ii) $x \in L_{\mathsf{pub},j,T}$, and (iii) $x \notin L_{\mathsf{pub},j,T'}$.

Winning conditions for $\mathcal{G}_{\mathsf{C.Prst}}^{\mathcal{A},\delta,\Delta,t_c}$.

The game outputs 1 iff for every moment T, we have that $\mathsf{VerifyPub}\big(\langle L_{\mathsf{pub},j,T}\rangle_{j\in[N_w]}, \mathsf{params}\big) = \mathsf{accept}$ and there is a (not necessarily honest) AB peer AB_j, such that at least one of the following conditions holds:

(CP.1). There are t_c+1 honest IC peers $\{P_{i_k}\}_{k\in[t_c+1]}$, an item x, and moments T', T'', such that (i) $T' \leq T''$, (ii) $x \in L_{\mathsf{pub},j,T'}$, and (iii) $x \notin L_{\mathsf{post},i_k,T''}$, for any $k \in [t_c+1]$.

(CP.2). There is an item x and moments T', T'' such that (i) $T' < T''$, (ii) $x \in L_{\mathsf{pub},j,T'}$, and (iii) $x \notin L_{\mathsf{pub},j,T''}$.

Winning conditions for $\mathcal{G}_{\theta-\mathsf{C.Live}}^{\mathcal{A},\delta,\Delta,t_c,t_w}$.

The game outputs 1 iff exists an honest user U, an item x and a moment T such that the following hold:

(CL.1). $\mathcal{A}$ provided $\mathcal{C}$ with the message (post, U, x) at global time $\mathsf{Clock} = T$.

(CL.2). No honest IC peer engages in the **Publishing** protocol during global time $\mathsf{Clock} \in [T, T+\theta]$.

(CL.3). Either of the following two is true:

(a) By global time $\mathsf{Clock} \leq T + \theta$, $\mathcal{C}$ did not obtain a value z such that $\mathsf{VerifyRec}(z, x, \mathsf{cr}_U, \mathsf{params}) = \mathsf{accept}$

(b) There is an AB peer $AB_j \notin L_{\mathsf{corr}}$, such that for any moment T_j, there exists a moment $T'_j \geq T_j$ such that $x \notin L_{\mathsf{pub},j,T'_j}$.

Fig. 1. Security games for (Confirmable) Persistence, and θ-Confirmable Liveness.

3.4 θ-Confirmable Liveness Definition

We define θ-Confirmable Liveness via a security game $\mathcal{G}_{\theta-\mathsf{C.Live}}^{\mathcal{A},\delta,\Delta,t_c,t_w}(1^\kappa,\mathbf{E})$ between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, where $\mathcal{A}$ statically corrupts up to t_c (resp. t_w) out-of the N_c (resp. N_w) total IC (resp. AB) peers, while $\mathcal{C}$ plays the role of SA and all peers and users that $\mathcal{A}$ does not corrupt. The adversary wins if it prevents the generation of a valid receipt for an item x or the eventual publishing of x, given that x has been submitted at least θ time prior to the nearest **Publishing** protocol execution. The game is described in detail in Fig. 1.

Definition 2 (θ-Confirmable Liveness). *Let κ be the security parameter, $N_c, N_w, t_c, t_w, \theta \in \mathbb{N}$, $\delta, \Delta \in [0,+\infty]$ and let* **BB** *be a BB system with N_c IC and N_w AB peers. We say that* **BB** has θ-Confirmable Liveness *for fault-tolerance thresholds (t_c, t_w), delay message bound δ, and synchronization loss bound Δ, if for every PPT adversary $\mathcal{A}$, it holds that* $\Pr\left[\mathcal{G}_{\theta-\mathsf{C.Live}}^{\mathcal{A},\delta,\Delta,t_c,t_w}(1^\kappa,\mathbf{E})=1\right]=\mathsf{negl}(\kappa)$.

4 The Culnane-Schneider (CS) BB system

In this section, we outline the CS BB system as presented in [14] adopted in our terminology, and analyze its security guarantees and weaknesses under the framework introduced in Sect. 3. The CS BB system comprises the setup authority SA, the users, the IC peers $P_1,\ldots,P_{N_c}$ and a single trusted AB (called WBB in [14]), i.e., $N_w = 1$. The fault-tolerance threshold on the number of corrupted IC peers, t_c, that CS requires is $t_c < N_c/3$ and $t_s + 1 = N_c - t_c$.

4.1 Overview of the CS BB System

Setup. Upon specifying the posting policy $\mathcal{P} = \big(\mathsf{Accept}, \mathsf{Select}(\cdot)\big)$, the SA provides all entities with the description of an EUFCMA-secure signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ and a (t_s, N_c)-EUFCMA-secure TSS $\mathsf{TSS} = (\mathsf{DistKeygen}$, $\mathsf{ShareSig}$, $\mathsf{ShareVerify}$, TVf, $\mathsf{Combine})$. Then, each IC peer P_i runs $\mathsf{KGen}(1^\kappa)$ to get a signing key sk_i and a verification key vk_i, while IC peers jointly execute $\mathsf{DistKeygen}(1^\kappa, t_s, N_c)$ to produce secret keys $\{\mathsf{tsk}_i\}_i$, implicitly defining tsk, and the corresponding public output $\mathsf{pk}, \{\mathsf{pk}_i\}_i$. Finally, the IC peers broadcast all public keys and every user U interacts with SA to obtain her private input cr_U.

The CS BB system runs in consecutive periods. Each period p is a time interval $[T_{\mathsf{begin},p}, T_{\mathsf{end},p}]$ between two fixed global time values $T_{\mathsf{begin},p}$ and $T_{\mathsf{end},p}$, and the end of a period matches the beginning of the next one. For each IC peer P_i, we denote by $B_{i,p}$ *the local record of* P_i including all items x recorded as posted and by $D_{i,p}$ *the database of received items* x together with other peers' signatures on them, for the period p. In the beginning of p, P_i sets $B_{i,p}, D_{i,p} \leftarrow \emptyset$.

Posting. If a user U wants to post item x during period p, then she broadcasts x to all IC peers, along with her credential cr_U. Upon receiving and verifying the validity of (x, cr_U), each peer P_i broadcasts a signature on (p, x, cr_U) under its singing key sk_i. When P_i receives $N_c - t_c$ valid signatures on (p, x, cr_U) (including its own) from $N_c - t_c$ different peers, it threshold signs (p, x) and sends it to U. Finally, when U receives $N_c - t_c \geq t_s + 1$ valid TSS shares from $N_c - t_c$ different peers, it combines them to obtain a threshold signature on (p, x), as her receipt. We define $\mathsf{VerifyRec}(\mathsf{rec}[x], x, \mathsf{cr}_U, \mathsf{params}) := \mathsf{TVf}_{\mathsf{pk}}((p, x), \mathsf{TSign}(\mathsf{tsk}, (p, x)))$.

Publishing. Given a period $p = [T_{\mathsf{begin},p}, T_{\mathsf{end},p}]$, all IC peers stop item recording and begin publishing their local records at a fixed time $T_{\mathsf{barrier},p} \in (T_{\mathsf{begin},p}, T_{\mathsf{end},p})$. The **Publishing** protocol includes two subprotocols: initially, the IC peers run an *Optimistic protocol* that results in the publishing of a BB record, if at least $N_c - t_c$ local BB records agree. We note that the Optimistic protocol always terminates successfully if all peers are honest. If the Optimistic protocol check fails, then IC peers engage in the *Fallback protocol*, where they exchange their databases of collected signatures for posted items. The Fallback protocol is essentially one round of the *Floodset agreement algorithm* [27, Sect. 6.2] with the following characteristic: if all users posted their items honestly, then Fallback need to run only once. Otherwise, as in standard Floodset, it needs to be executed up to $N_c - t_c + 1$ times in the synchronous setting.

When consensus is reached, the IC peers provide the AB with their records along with corresponding TSS shares. The AB sets the agreed record as its view for period p along with the reconstructed TSS signature from the collected shares. The total view of AB at some moment T, denoted by $L_{\mathsf{pub},T}$, is the union of the agreed and published BB records for all periods preceding moment T.

4.2 Attacking the Liveness of the CS BB System

As informally argued in [13, Sect. 8] (the full version of [14]), the liveness in CS can be achieved if one of the following conditions hold: (1) all the peers are following the protocol honestly and are online, (2) a threshold of $t_c < N_c/3$ peers is malicious, but all users are honest, or (3) the more general condition that not all users are honest and *the malicious peers may choose any database in their capability, but do not change their database once it has been fixed, and will not send different databases to different peers.* The argument is that one can easily detect in practice if malicious peers send different databases to different peers.

We demonstrate an attack against the Confirmable Liveness of CS in our framework. Although our attack falls outside the threat model of [14], it reveals that the presumed "fear of detection" that justifies the said threat model, and especially the more general condition (3) described above, is not rigorously addressed. In particular, we show that the liveness adversary may choose to split the honest peers into two groups, and yet not be detected by being consistent w.r.t. to the peers in the same group. This way, the adversary manages a liveness breach, while the honest IC peers cannot detect the attack relying on the protocol guidelines and their local views. As a result, our attack shows that the original description of CS must be enhanced with an explicit detection mechanism against any deviation from the IC consensus protocol specifications, in order for the threat model in [14] to be properly justified. On the other hand, as we describe in Sect. 5 and prove in Sect. 6, enhancing CS with our novel **Publishing** protocol completely overcomes such issues, by achieving Confirmable Liveness even against a general Byzantine adversary.

Description of the Liveness Attack. Our attack works under fault tolerance threshold $N_c > 3t_c$, as required in [14], and consists of the steps below.

STEP 1: Let p be a period where the set of honestly posted items is non-empty. For simplicity, we assume that there is a single honest user U_h who broadcasts x_h to all IC peers P_i, $i \in [N_c]$, and obtains a valid receipt $\mathsf{rec}[x_h]$.

STEP 2: A malicious user U_c deviates from broadcasting and sends x_c to all t_c corrupted peers and $N_c - 2t_c$ honest peers. Denote the latter set of honest $N_c - 2t_c$ peers by $\mathcal{H}_{\mathsf{in}}$. The malicious peers engage in the **Posting** protocol by interacting only with the peers in $\mathcal{H}_{\mathsf{in}}$. Observe that even if t_c honest peers do not participate in the post request of x_c, the collaboration of $t_c+(N_c-2t_c) = N_c-t_c$ peers is enough so that U_c obtains a valid receipt $\mathsf{rec}[x_c]$, yet $(p, x_c) \in B_{i,p}$ only for honest peers $P_i \in \mathcal{H}_{\mathsf{in}}$. Denote by $\mathcal{H}_{\mathsf{out}}$ the t_c honest peers s.t. $x_c \notin B_{i,p}$.

STEP 3: Another malicious user $\hat{U}_c$ deviates from broadcasting and, like U_c, sends item $\hat{x}_c$ to all t_c corrupted peers and the $N_c - 2t_c$ honest peers in $\mathcal{H}_{\mathsf{in}}$. However, now the malicious peers do not engage in the **Posting** protocol, so the peers in $\mathcal{H}_{\mathsf{in}}$ do not suffice for a receipt for $\hat{x}_c$.

STEP 4: When **Publishing** protocol starts, the honest peers in $\mathcal{H}_{\mathsf{in}}$ and $\mathcal{H}_{\mathsf{out}}$ engage in the Optimistic protocol by sending their signed local records $\mathcal{R}_h^c := \{(p, x_h), (p, x_c)\}$ and $\mathcal{R}_h := \{(p, x_h)\}$ respectively. From their side, the malicious peers sign their records as $\mathcal{R}_h^{c,\hat{c}} := \{(p, x_h), (p, x_c), (p, \hat{x}_c)\}$. As a result, none of the three records $\mathcal{R}_h$, $\mathcal{R}_h^c$ and $\mathcal{R}_h^{c,\hat{c}}$ is signed by at least $N_c - t_c$ peers (recall that $|\mathcal{H}_{\mathsf{in}}| = N_c - 2t_c$ and $|\mathcal{H}_{\mathsf{out}}| = t_c$). Therefore, the malicious peers force all honest peers to engage in the Fallback protocol.

STEP 5: During Fallback, all honest peers exchange their collection of signatures. At this step, each peer in $\mathcal{H}_{\mathsf{in}}$ sends to each peer in $\mathcal{H}_{\mathsf{out}}$ (i) its signature on $(p, x_c), (p, x_h)$ and $(p, \hat{x}_c)$ and (ii) the t_c signatures on (p, x_c) that it received from the malicious peers. This way, each peer in $\mathcal{H}_{\mathsf{out}}$ receives $(N_c-2t_c)+t_c = N_c-t_c$ signatures on (p, x_c) but only N_c-2t_c signatures on $(p, \hat{x}_c)$, so it updates its local record to $\mathcal{R}_h^c$. Malicious peers send their signatures on $(p, x_c), (p, x_h)$ and $(p, \hat{x}_c)$ only to the peers in $\mathcal{H}_{\mathsf{in}}$. Therefore, each peer collects $(N_c - 2t_c) + t_c = N_c - t_c$ signatures on $(p, \hat{x}_c)$ and updates its local record to $\mathcal{R}_h^{c,\hat{c}}$.

STEP 6: When the Fallback round above is completed, all peers restart the Optimistic protocol. However, now the peers in $\mathcal{H}_{\mathsf{in}}$ and $\mathcal{H}_{\mathsf{out}}$ send their signed local records $\mathcal{R}_h^{c,\hat{c}}$ and $\mathcal{R}_h^c$ respectively. The malicious peers resend their records $\mathcal{R}_h^{c,\hat{c}}$ only to the peers in $\mathcal{H}_{\mathsf{in}}$, which now have N_c-t_c signatures on $\mathcal{R}_h^{c,\hat{c}}$. Thus, they finalize their engagement in the **Publishing** protocol for period p by sending their TSS shares for $\mathcal{R}_h^{c,\hat{c}}$ to the AB.

STEP 7: After forcing the peers in $\mathcal{H}_{\mathsf{in}}$ to termination, the malicious peers become inert. This causes the peers in $\mathcal{H}_{\mathsf{out}}$ to remain pending for a new Fallback round that no other peer will follow. Moreover, the AB can not obtain $N_c - t_c$ TSS shares on some agreed record, and thus it can not publish anything. This violates the property (bb.2) in [14] (expressed via condition (L.3) in Fig. 1), which dictates that since x_h is an honestly posted item that has a receipt, it must be published to the AB. Thus, liveness is breached.

4.3 TSS Fault-Tolerance Requirements for Confirmable Persistence

In [14] no concrete recommendations are given for which TSS to use. For liveness to be achieved, TSS should be robust, i.e., malicious peers cannot block signature creation. However, robustness is feasible only if $t_s < N_c/2$ [19], which contradicts the CS requirement $t_c < N_c/3$ and $t_s + 1 = N_c - t_c > 2N_c/3$. Given that $t_s < N_c/2$, we can still prove the CS BB system to achieve Confirmable Persistence, but for a smaller bound of $t_c < N_c/4$. This bound is tight, in the sense that if $t_c \geq N_c/4$, then there exists an attack. Thorough treatment of this issue is provided in the full version [24].

5 A New Publishing Protocol for the CS BB System

We present a new **Publishing** protocol that, when combined with the CS **Posting** protocol, results in a BB system that achieves Confirmable Liveness in partially synchronous and Persistence in asynchronous model, against a general Byzantine adversary, assuming a threshold of $t_c < N_c/3$ corrupted IC peers. Persistence can also be Confirmable, if we distribute the AB subsystem such that no more than $t_w < N_w/2$ out of the N_w AB peers are corrupted, as in [10]. Namely, the distributed AB runs as a replication service; data posting is done by broadcasting to all AB peers, while data reading is done by honest majority.

The public parameters params include the identities of the IC and AB peers, the description of $\mathsf{DS}, \mathsf{TSS}$ (cf. Sect. 2), a *collision resistant hash function (CRHF)* $H_\kappa(\cdot)$, and all public and verification keys. All peers know consecutive periods $p = [T_{\mathsf{begin},p}, T_{\mathsf{end},p}]$, as well as the following moments per period p: (a) a moment $T_{\mathsf{barrier},p} \in (T_{\mathsf{begin},p}, T_{\mathsf{end},p})$, when item collection stops and the **Publishing** protocol is initiated; (b) a moment $T_{\mathsf{publish},p} \in (T_{\mathsf{barrier},p}, T_{\mathsf{end},p})$, where the AB peers publish their records for period p, and (c) a moment $T_{\mathsf{request},p} \in (T_{\mathsf{barrier},p}, T_{\mathsf{publish},p})$, where IC peers force exchange of information to finalize their records. For each period p, the phases of the **Publishing** protocol are as follows:

- **Initialization** phase: each IC peer P_i initializes the following vectors:

  (i) Its *direct view* of local records, denoted by $\mathsf{View}_{i,p} := \langle \tilde{B}_{i,1,p}, \ldots, \tilde{B}_{i,N_c,p} \rangle$: namely, it sets $\tilde{B}_{i,j,p} \leftarrow \perp$, for $j \neq i$, and $\tilde{B}_{i,i,p} \leftarrow B_{i,p}$.

  (ii) For every $j \in [N_c] \setminus \{i\}$, its *indirect view* of local records as provided by P_j, denoted by $\mathsf{View}_{i,j,p} := \langle \tilde{B}^i_{j,1,p}, \ldots, \tilde{B}^i_{j,N_c,p} \rangle$, by setting $\mathsf{View}_{i,j,p} \leftarrow \langle \perp, \ldots, \perp \rangle$.

(iii) A variable vector $\langle b_{i,1}, \ldots, b_{i,N_c} \rangle$, where $b_{i,j}$ is a value in $\{?, 0, 1\}$ that expresses *the opinion of P_i on the validity of P_j's behavior*. Initially, $b_{i,i}$ is fixed to 1, while for $j \neq i$, $b_{i,j}$ is set to the "pending" value '?'. When P_i fixes $b_{i,j}$ to 1/0 for all $j \in [N_c]$, it engages in the **Finalization** phase described shortly.

(iv) A vector $\langle d_{i,1}, \ldots, d_{i,N_c} \rangle$, where $d_{i,j}$ is *the number of P_i's (direct or indirect) views that agree on P_j's record*. Initially, $d_{i,j} = 0$, for $j \neq i$, and $d_{i,i} = 1$.

■ **Collection** phase: upon initialization, P_i signs its local record $B_{i,p}$, followed by a tag RECORD, and broadcasts $\big((\text{RECORD}, B_{i,p}), \mathsf{Sig}_{\mathsf{sk}_i}(\text{RECORD}, B_{i,p})\big)$ to all IC peers. Then, P_i updates its direct and indirect views of other IC peers' records and fixes its opinion bit for their behavior, depending on the number of consistent signed messages it receives on each peer's record. In particular,

– *When P_i receives a message $\big((\text{RECORD}, R_{i,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, R_{i,j,p})\big)$ signed by peer P_j that was never received before*, then it acts as follows: if $R_{i,j,p}$ is formatted as a non-$\perp$ record and the "opinion" bit $b_{i,j}$ is not fixed (i.e. $b_{i,j} =$ '?'), then it checks if $\mathsf{Vf}_{\mathsf{pk}_j}\big((\text{RECORD}, R_{i,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, R_{i,j,p})\big) = 1$. If the latter holds, then P_i operates according to either of the following two cases:

1. If $\tilde{B}_{i,j,p} \neq \perp$, then it marks P_j as malicious, that is, it sets $\tilde{B}_{i,j,p} \leftarrow \perp$ and fixes $b_{i,j}$ to 0. Observe that since P_j is authenticated (except from some $\mathsf{negl}(\kappa)$ error), it is safe for P_i to mark P_j as malicious, as an honest peer would never send two different versions of its local records.
2. If $\tilde{B}_{i,j,p} = \perp$, then P_i updates $\mathsf{View}_{i,p}$ as $\tilde{B}_{i,j,p} \leftarrow R_{i,j,p}$, and $\mathsf{View}_{i,j,p}$ as $\tilde{B}^i_{j,j,p} \leftarrow R_{i,j,p}$ and increases the $d_{i,j}$ by 1. Next, it signs and re-broadcasts to all IC peers the received message in the format $\big(V_{i,j}, \mathsf{Sig}_{\mathsf{sk}_i}(V_{i,j})\big)$, where $V_{i,j} := \big((\text{VIEW}, j), ((\text{RECORD}, \tilde{B}_{i,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, \tilde{B}_{i,j,p}))\big)$. Upon fixing $b_{i,j}$ to 1/0, P_i ignores any further message for the record of P_j.

– *When P_i receives a message $\big(V_{k,j}, \mathsf{Sig}_{\mathsf{sk}_k}(V_{k,j})\big)$ signed by peer P_k for some peer P_j different than P_i and P_k, where $V_{k,j} = \big((\text{VIEW}, j), ((\text{RECORD}, R_{k,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, R_{k,j,p}))\big)$, and the message was never received before*, then it acts as follows: if $R_{k,j,p}$ is formatted as a non-$\perp$ record and $b_{i,j} =$ '?', then it executes verification $\mathsf{Vf}_{\mathsf{pk}_k}(V_{k,j}, \mathsf{Sig}_{\mathsf{sk}_k}(V_{k,j}))$. If $\mathsf{Vf}_{\mathsf{pk}_k}(V_{k,j}, \mathsf{Sig}_{\mathsf{sk}_k}(V_{k,j})) = 1$, then P_i operates according to either of the following two cases:

1. If $\mathsf{Vf}_{\mathsf{pk}_j}\big((\text{RECORD}, R_{i,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, R_{i,j,p})\big) = 0$ or $\tilde{B}^i_{k,j,p} \neq \perp$, then P_i sets $\tilde{B}_{i,k,p} \leftarrow \perp$, fixes the bit $b_{i,k}$ to 0[2].
2. If $\mathsf{Vf}_{\mathsf{pk}_j}\big((\text{RECORD}, R_{i,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, R_{i,j,p})\big) = 1$ and $\tilde{B}^i_{k,j,p} = \perp$, then P_i updates $\mathsf{View}_{i,k,p}$ by setting $\tilde{B}^i_{k,j,p} \leftarrow R_{k,j,p}$. and $\mathsf{View}_{i,p}$ as shown below:

[2] Observe that it is safe for P_i to mark P_k as a malicious, since an honest P_k would neither send two non-$\perp$ views for P_j, nor accept an invalid signature from P_j.

(C.1). If for every $k' \in [N_c] \setminus \{i\}$ such that $\tilde{B}^i_{k',j,p} \neq \perp$, it holds that $\tilde{B}^i_{k',j,p} = \tilde{B}^i_{k,j,p} := \tilde{B}^i_{j,p}$ (i.e. all non-$\perp$ records for j agree on some record $\tilde{B}^i_{j,p}$), then it increases the value of $d_{i,j}$ by 1. Moreover, if $d_{i,j} = t_c + 1$, (i.e., there are $t_c + 1$ matching non-$\perp$ records) and $\tilde{B}_{i,j,p} = \perp$, then it updates as $\tilde{B}_{i,j,p} \leftarrow \tilde{B}^i_{j,p}$ and fixes the bit $b_{i,j}$ to 1.
(C.2). If there is a $k' \in [N_c]$ such that $\tilde{B}^i_{k',j,p} \neq \perp$ and $\tilde{B}^i_{k,j,p} \neq \tilde{B}^i_{k',j,p}$, then it updates as $\tilde{B}_{i,j,p} \leftarrow \perp$ and fixes the bit $b_{i,j}$ to 0.

In either case, upon fixing $b_{i,j}$, P_i ignores any further message for P_j's record[3].

– *When its local clock* $\mathsf{Clock}[P_i]$ *reaches* $T_{\mathsf{request},p}$, P_i broadcasts a request message $\big((\text{REQUEST\_VIEW}, j), \mathsf{Sig}_{\mathsf{sk}_i}(\text{REQUEST\_VIEW}, j)\big)$, for every P_j that it has not yet fixed the opinion bit $b_{i,j}$. This step is executed to ensure that P_i *will eventually fix its opinion bits for all IC peers.* Upon receiving P_i's request, P_k replies with a signature for a response message $\big(W_{k,j}, \mathsf{Sig}_{\mathsf{sk}_k}(W_{k,j})\big)$, where $W_{k,j} := \big((\text{RESPONSE\_VIEW}, j), ((\text{RECORD}, R_{k,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, R_{k,j,p}))\big)$. Note that here $R_{k,j,p}$ may be $\perp$, reflecting the P_k's lack of direct view for P_j's record. For every P_j that P_i has broadcast $\big((\text{REQUEST\_VIEW}, j), \mathsf{Sig}_{\mathsf{sk}_i}(\text{REQUEST\_VIEW}, j)\big)$, P_i waits until it collects $N_c - t_c - 1$ distinct valid signed responses. During this wait, it ignores any message in a format other than $\big(W_{k,j}, \mathsf{Sig}_{\mathsf{sk}_k}(W_{k,j})\big)$ or $\big((\text{REQUEST\_VIEW}, j), \mathsf{Sig}_{\mathsf{sk}_k}(\text{REQUEST\_VIEW}, j)\big)$. When $N_c - t_c - 1$ distinct valid responses are received, it parses the collection of the $N_c - t_c - 1$ responses and its current direct view of P_j's record, $\tilde{B}_{i,j,p}$, to update $\tilde{B}_{i,j,p}$ and fix $b_{i,j}$ as follows:

(R.1). If $\tilde{B}_{i,j,p} \neq \perp$, and all responses for non-$\perp$ records are at least t_c and all match $\tilde{B}_{i,j,p}$, then P_i fixes $b_{i,j}$ to 1.
(R.2). If $\tilde{B}_{i,j,p} = \perp$, and all responses for non-$\perp$ records are at least $t_c + 1$ and all refer to the same record denoted as $\tilde{B}^i_{j,p}$, then P_i sets $\tilde{B}_{i,j,p} \leftarrow \tilde{B}^i_{j,p}$ and fixes $b_{i,j}$ to 1.
(R.3). Otherwise, P_i sets $\tilde{B}_{i,j,p} \leftarrow \perp$ and fixes $b_{i,j}$ to 0.

In any case, upon fixing $b_{i,j}$, P_i ignores any further message for P_j's record[4]. At the end of the **Collection** phase, P_i will have fixed $b_{i,j}$ for all $j \in [N_c]$.

■ **Finalization** phase: having fixed $b_{i,1} \dots, b_{i,N_c}$ and updated its direct view $\mathsf{View}_{i,p} := \langle \tilde{B}_{i,1,p}, \dots, \tilde{B}_{i,N_c,p} \rangle$, peer P_i proceeds as follows: for every pair $(p, x) \in \bigcup_{j : \tilde{B}_{i,j,p} \neq \perp} \tilde{B}_{i,j,p}$, P_i defines the set $N_{i,p}(x)$ that denotes the number of IC peers that, according to its view, have included (p, x) in their records. Formally, we

[3] The security of DS ascertains P_i that with $1 - \mathsf{negl}(\kappa)$ probability, only if P_j is malicious, two non-equal records can be valid under P_j's verification key. Thus, in case (C.2), P_i can safely fix the bit $b_{i,j}$ to 0.

[4] Since there are $N_c - t_c \geq t_c + 1$ honest peers, P_i will obtain at least $t_c + 1$ all matching non-$\perp$ views for every honest' peers record (including its own). Thus, in case (R.3), P_i can safely fix $b_{i,j}$ to 0 if it receives inconsistent non-$\perp$ views or less than $t_c + 1$ matching non-$\perp$ views for P_j.

write $N_{i,p}(x) := \#\{j \in [N_c] : (p, x) \in \tilde{B}_{i,j,p}\}$. Then, P_i updates its original record $B_{i,p}$ as follows:

(F.1). If $(p, x) \notin B_{i,p}$, but $N_{i,p}(x) \geq t_c + 1$, then it adds (p, x) in $B_{i,p}$.
(F.2). If $(p, x) \in B_{i,p}$, but $N_{i,p}(x) < t_c + 1$, then it removes (p, x) from $B_{i,p}$.

In any other case, $B_{i,p}$ becomes unchanged[5]. As shown in Theorem 2, at the end of the **Finalization** phase, all honest peers have included all honestly posted items for which a receipt has been generated in their local records. Then, they advance to the **Publication** phase described below.

■ **Publication** phase: each peer P_i threshold signs its record $B_{i,p}$, as it has been updated during the **Finalization** phase, by threshold signing each item in $B_{i,p}$ individually. Formally, $\mathsf{ShareSig}(\mathsf{tsk}_i, (p, B_{i,p})) := \bigcup_{(p,x)\in B_{i,p}} \mathsf{ShareSig}(\mathsf{tsk}_i, (p, x))$. Then, P_i broadcasts the message $\big((p, B_{i,p}), \mathsf{ShareSig}(\mathsf{tsk}_i, (p, B_{i,p}))\big)$ to all peers $AB_1, \ldots, AB_{N_w}$ of the AB subsystem.

In turn, each peer AB_j, $j \in [N_w]$ receives and records threshold signature shares for posted items. For every item (p, x) that AB_j receives $N_c - t_c$ valid signatures shares $(k, \sigma_k)_{k\in S}$, where S is a subset of $N_c - t_c$ IC peers, it adds (p, x) to its record $B_p[j]$, initialized as empty, and computes a TSS signature on (p, x) as $\mathsf{TSign}(\mathsf{tsk}, (p, x)) \leftarrow \mathsf{Combine}\big(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c}, (p, x), (k, \sigma_k)_{k\in S}\big)$. Upon finalizing $B_p[j]$, AB_j executes the following steps:

1. It sets $\mathsf{TSign}(\mathsf{tsk}, (p, B_p[j])) := \bigcup_{(p,x)\in B_p[j]} \mathsf{TSign}(\mathsf{tsk}, (p, x))$ and when its local clock $\mathsf{Clock}[AB_j]$ reaches $T_{\mathsf{publish},p}$, it publishes the signed record
$$\mathsf{ABreceipt}[p, B_p[j]] := \big((p, B_p[j]), \mathsf{TSign}(\mathsf{tsk}, (p, B_p[j]))\big).$$
2. By the time that the period p ends (i.e., $\mathsf{Clock}[AB_j] = T_{\mathsf{end},p}$), for $k \in [N_w] \setminus \{j\}$, it performs a read operation on AB_k and reads its record for period p denoted by $B_p[j, k]$ (possibly empty). Then, it publishes the hash $H_\kappa\big(B_p[j, k]\big)$ of the read record.

The VerifyPub Algorithm. Let $\mathsf{Prec}[p]$ be the set of all periods preceding p. The total view of AB_j at some moment T during period p, denoted by $L_{\mathsf{pub},j,T}$, is the union of the published BB records $B_{\tilde{p}}[j]$ for all periods $\tilde{p} \in \mathsf{Prec}[p]$.

On input $\big(\langle L_{\mathsf{pub},j,T}\rangle_{j\in[N_w]}, \mathsf{params}\big)$, the algorithm $\mathsf{VerifyPub}$ outputs accept iff for every $j \in [N_w]$ and every $\tilde{p} \in \mathsf{Prec}[p]$ the following hold:

(a) More than $N_w/2$ AB peers that agree on the consistency of the data that AB_j publishes (including AB_j). Formally, there is a subset $\mathcal{I}_j \subseteq [N_w]$ such that $|\mathcal{I}_j| > N_w/2$ and $\forall k \in \mathcal{I}_j \setminus \{j\} : H_\kappa\big(B_{\tilde{p}}[k, j]\big) = H_\kappa\big(B_{\tilde{p}}[j]\big)$.
(b) For every $(\tilde{p}, x) \in B_{\tilde{p}}[j]$, it holds that $\mathsf{TVf}\big(\mathsf{pk}, (\tilde{p}, x), \mathsf{TSign}(\mathsf{tsk}, (\tilde{p}, x))\big) = 1$.

[5] In case (F.2), removal is a safe action for P_i, as every honestly posted item for which a receipt has been generated, is stored in the records of at least $N_c - 2t_c \geq t_c + 1$ honest peers during the **Posting** protocol. Thus, $N_{i,p}(x) < t_c + 1$ implies that either (i) (p, x) was maliciously posted, or (ii) a receipt for (p, x) was not generated.

An item belongs in the published data of the whole AB system by moment T, denoted by $L_{\mathsf{pub},T}$, if it appears on more than half of the AB peers. Formally,

$$L_{\mathsf{pub},T} := \bigcup_{\tilde{p}\in\mathsf{Prec}[p]} \Big\{(\tilde{p},x)\Big|\#\{j \in [N_w] : (\tilde{p},x) \in B_{\tilde{p}}[j]\} > N_w/2\Big\}.$$

Complexity of the New Publishing Protocol. Our protocol has a constant number of rounds per period, where the size of transmitted messages is equal to the signature on records of items posted on the said period. In particular, the **Collection** phase has cubic ($\sim(N_c)^3$) communication complexity (the IC peers exchange their views), while the **Publication** phase has quadratic ($\sim N_c \cdot N_w$) communication complexity (the IC peers broadcast their updated records to the AB peers). Overall, the complexity of the new **Publishing** protocol matches the one of the original CS system (cf. Sect. 4.1), as in general, the Floodset algorithm must run in $N_c - t_c + 1$ rounds, where in each round a full quadratic communication for mutual information exchange is required.

6 Properties of the New BB System

In this section, we analyze the security of the BB system that comprises the **Setup** and the **Posting** protocol of CS combined with our novel **Publishing** protocol described in Sect. 5. For simplicity, we will refer to this BB system as the system described in Sect. 5. We write $T_{\mathcal{B}}$ to denote the running time of algorithm $\mathcal{B}$, omitting parameterization by the security parameter κ for brevity. The parameters N_c, t_c are considered polynomial in κ. In our setting, we assume that the message delivery delay δ and the synchronization loss bound Δ are small enough with respect to the protocol steps and the intervals $[T_{\mathsf{begin},p}, T_{\mathsf{barrier},p}]$, $[T_{\mathsf{barrier},p}, T_{\mathsf{request},p}]$, $[T_{\mathsf{request},p}, T_{\mathsf{publish},p}]$, $[T_{\mathsf{publish},p}, T_{\mathsf{end},p}]$ that determine phase switching in each period p. We consider that this restriction does not effectively violate partial synchrony, as the actual δ, Δ need not to be known to the IC peers for executing the protocol. Due to space limitations, we only provide the theorem statement and leave proofs for the full version [24]. In Table 1, we provide a brief comparison between the original CS BB system and its improved variant over the new **Publishing** protocol. For better comparison, we also consider CS BB in the setting where the AB is distributed.

Theorem 1 (Confirmable Persistence). *Let $N_c, N_w, t_c, t_w, t_s \in \mathbb{N}$, such that (a) $t_c < N_c/3$, (b) $t_w < N_w/2$ and (c) $t_s \geq N_c - t_c - 1$, and let $\delta = \Delta = \infty$. Let TSS be a (t_s, N_c)-EUFCMA-secure TSS and H_κ be a CRHF. Then, the BB system described in Sect. 5 with N_c IC peers and N_w AB peers over TSS and H_κ achieves (i) Persistence for tolerance thresholds (t_c, N_w), and (ii) Confirmable Persistence for tolerance thresholds (t_c, t_w).*

Theorem 2 (Confirmable Liveness). *Let $N_c, N_w, t_c, t_w, t_s \in \mathbb{N}$ such that (a) $t_c < N_c/3$, (b) $t_w < N_w$, and (c) $t_c \leq t_s < N_c - t_c$, and $\delta, \Delta \in \mathbb{R}_{\geq 0}$. Let DS be an*

EUFCMA-secure signature scheme and TSS *be a robust and* (t_s, N_c)*-EUFCMA-secure TSS. Then, the BB system described in Sect.* 5 *with* N_c *IC peers and* N_w *AB peers over* DS *and* TSS *achieves* θ*-Confirmable Liveness for fault tolerance thresholds* (t_c, t_w)*, delay message bound* δ *and synchronization loss bound* Δ*, and for every* $\theta \geq \Delta + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}$.

Table 1. Comparison of CS BB and the new BB with N_c IC peers and N_w AB peers.

BB	Complexity	Persistence	Con. Persistence	Con. Liveness
[14]	$\sim(N_c)^3$	Asynchronous $t_c < \frac{N_c}{3}$, $t_w \leq N_w$	Asynchronous $t_c < \frac{N_c}{3}$, $t_w \leq N_w$	Synchronous $t_c = 0$, $t_w < \frac{N_w}{2}$
This work	$\sim(N_c)^3$	Asynchronous $t_c < \frac{N_c}{3}$, $t_w \leq N_w$	Asynchronous $t_c < \frac{N_c}{3}$, $t_w < \frac{N_w}{2}$	Part. Synchronous $t_c < \frac{N_c}{3}$, $t_w < \frac{N_w}{2}$

References

1. Adida, B.: Helios: web-based open-audit voting. In: USENIX (2008)
2. Aumann, Y., Lindell, Y.: Security against covert adversaries: efficient protocols for realistic adversaries. J. Cryptol. **23**(2), 281–343 (2010)
3. Benaloh, J.: Verifiable secret-ballot elections. Ph.D. thesis. Yale University (1987)
4. Benaloh, J., et al.: STAR-Vote: a secure, transparent, auditable, and reliable voting system. In: EVT/WOTE 2013 (2013)
5. Burton, C., et al.: Using Prêt à voter in Victoria state elections. In: EVT/WOTE (2012)
6. Burton, C., Culnane, C., Schneider, S.: vVote: verifiable electronic voting in practice. IEEE Secur. Priv. **14**(4), 64–73 (2016)
7. Chaum, D.: SureVote: technical overview. In: WOTE (2001)
8. Chaum, D., et al.: Scantegrity: end-to-end voter-verifiable optical-scan voting. IEEE Secur. Priv. **6**(3), 40–46 (2008)
9. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.C., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005). https://doi.org/10.1007/11555827_8
10. Chondros, N., et al.: D-DEMOS: a distributed, end-to-end verifiable, internet voting system. In: ICDCS (2016)
11. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: EUROCRYPT, pp. 103–118 (1997)
12. Culnane, C., Ryan, P.Y.A., Schneider, S.A., Teague, V.: vVote: a verifiable voting system. ACM Trans. Inf. Syst. Secur. **18**(1), 3:1–3:30 (2015)
13. Culnane, C., Schneider, S.: A peered bulletin board for robust use in verifiable voting systems. CoRR abs/1401.4151 (2014)
14. Culnane, C., Schneider, S.A.: A peered bulletin board for robust use in verifiable voting systems. In: CSF (2014)

15. Dini, G.: A secure and available electronic voting service for a large-scale distributed system. Future Gener. Comput. Syst. **19**(1), 69–85 (2003)
16. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. ACM **35**(2), 288–323 (1988)
17. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57220-1_66
18. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
19. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust and efficient sharing of RSA functions. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 157–172. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_13
20. Hauser, S., Haenni, R.: A generic interface for the public bulletin board used in UniVote. In: CeDEM (2016)
21. Heather, J., Lundin, D.: The append-only web bulletin board. In: Degano, P., Guttman, J., Martinelli, F. (eds.) FAST 2008. LNCS, vol. 5491, pp. 242–256. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01465-9_16
22. Heiberg, S., Willemson, J.: Verifiable internet voting in Estonia. In: EVOTE (2014)
23. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES (2005)
24. Kiayias, A., Kuldmaa, A., Lipmaa, H., Siim, J., Zacharias, T.: On the security properties of e-voting bulletin boards. Cryptology ePrint Archive, Report 2018/567 (2018)
25. Kiayias, A., Zacharias, T., Zhang, B.: End-to-end verifiable elections in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 468–498. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_16
26. Krummenacher, R.: Implementation of a web bulletin board for e-voting applications. Institute for Internet Technologies and Applications (2010)
27. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann, Burlington (1996)
28. Peters, R.A.: A secure bulletin board. Master's thesis. Eindhoven UT (2005)
29. Reiter, M.K.: The Rampart toolkit for building high-integrity services. In: Birman, K.P., Mattern, F., Schiper, A. (eds.) Theory and Practice in Distributed Systems. LNCS, vol. 938, pp. 99–110. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60042-6_7
30. Sandler, D., Wallach, D.S.: Casting votes in the auditorium. In: EVT (2007)