THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

On the Security Properties of e-Voting Bulletin Boards

Citation for published version:
Kiayias, A, Kuldmaa, A, Lipmaa, H, Siim, J & Zacharias, T 2018, On the Security Properties of e-Voting Bulletin Boards. in *11th Conference on Security and Cryptography for Networks (SCN 2018).* Lecture Notes in Computer Science, vol. 11035, Springer, Cham, Amalfi, Italy, pp. 505-523, 11th Conference on Security and Cryptography for Networks, Amalfi, Italy, 5/09/18. https://doi.org/10.1007/978-3-319-98113-0_27

Digital Object Identifier (DOI):
10.1007/978-3-319-98113-0_27

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
11th Conference on Security and Cryptography for Networks (SCN 2018)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 23. Apr. 2024

On the Security Properties of e-Voting Bulletin Boards

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

[1] University of Edinburgh, UK
[2] University of Tartu, Estonia
[3] Cybernetica-Smartmatic CCEIV, Estonia
[4] STACC, Estonia

akiayias@inf.ed.ac.uk, annabell.kuldmaa@gmail.com, helger.lipmaa@gmail.com, jannosiim@gmail.com, tzachari@inf.ed.ac.uk

Abstract. In state-of-the-art e-voting systems, a bulletin board (BB) is a critical component for preserving election integrity and availability. Although it is common in the literature to assume that a BB is a centralized entity that is trusted, in the recent works of Culnane and Schneider [CSF 2014] and Chondros *et al.* [ICDCS 2016], the importance of removing BB as a single point of failure has been extensively discussed. Motivated by these works, we introduce a framework for the formal security analysis of the BB functionality. 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 are *confirmable*, in the sense that any malicious behavior of the BB system can be detected via a verification mechanism.

As a case study for our framework, we analyze the BB system of Culnane and Schneider and point out its weaknesses. We demonstrate an attack revealing that the said system does not achieve Confirmable Liveness, even in the case where the adversary is computationally bounded and covert, i.e., it may deviate from the protocol specification but does not want to be detected. 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 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 peers both for Persistence and Confirmable Liveness, against a computationally bounded general Byzantine adversary.

Keywords: Bulletin board, e-voting, liveness, persistence

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. Furthermore, besides e-voting, verifiability of the execution is a desired feature in several other cases, such as auctions and blockchain transactions, to name a few.

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 traffic 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. This matter has been extensively studied by Barak *et al.* [5], where it is shown that parties communicating via non-authenticated channels are unavoidably prone to such "isolation" attacks.

When it comes to voting, consistency may be realized in various ways depending on the election setting. For instance, 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. Depending on the implementation, the IC and the AB could be distributed or centralized, or even managed by the same physical entity. Nonetheless, the above generic description typifies the way BB's are treated in the e-voting literature.

By the discussion so far, it is of high importance that the BB functionality implemented by the IC and AB should function as an *immutable database*, so that once submitted, items could not be erased or changed. The desired features of such a database include: (a) the ability to authenticate item contributors, (b) distributed operation so as to protect against attacks on availability, (c) a predetermined time-span where item submission is enabled, (d) resilience to any 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 [6] assumes BBs with append-only write operations for auditing, also stressing out that "implementing such bulletin boards may be a problem unto itself." Subsequently, most verifiable e-voting systems proposed (e.g. [23,17,13,31,15,2,14,10,7,32]) 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 [39,21,38,29,33], the most concrete examples include the BB applied in the vVote e-voting system [20] (cf. [18,11]) and the BB of the D-DEMOS Internet-voting (i-voting) system [16]. In all these cases, the introduced BB was either an integral part of a specific e-voting system [21,16], or, even though modular, lacked formal treatment under a comprehensive security model [39,38,33,29,20].

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 [20], 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, [20] prescribes a liveness guarantee of the eventual publishing of all submitted items for which a *receipt* of correct recording has been generated. Taking a step further, we introduce a framework for the formal study of the BB concept and its security. Our framework is inspired by (i) the notion of a *robust public transaction ledger* (RPTL) defined by Garay *et al.* [24] and (ii) the security model presented by Chondros *et al.* [16], thereby utilizing the connection between blockchain and BB systems, which, albeit being folklore, was never formally exploited. We formalize 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 [24], we divide security into two 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 [20], which we refer to as the CS BB system, that utilizes standard signature and threshold signature schemes (TSS) as cryptographic building blocks. In the threat model of [20], 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.

Our findings reveal that CS is not secure in our framework for the $< N_c/3$ threshold. Specifically, we demonstrate an explicit attack showing that CS with N_c IC peers does not achieve Confirmable Liveness. Our attack falls outside the threat model [20] but raises an interesting discussion about its plausibility. In particular, the threat model of [20] relies on a presumed "fear of detection" (cf. the full version of [20], [19, Section 8]), to exclude certain adversarial protocol deviations in the IC subsystem. Nevertheless, such covert type of security reasoning, cf. [3], is never formalized or implemented in [19] and in fact, as our attack demonstrates, the detection of protocol deviation is impossible by the IC peers themselves given their local protocol view.

A second, though less crucial, finding for the security of CS concerns its Confirmable Persistence guarantees. Namely, we show that in order 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 that care should be taken for the choice of the TSS construction. By describing a specific attack, we show that when the TSS robustness condition analyzed in [26] is invoked for liveness, then a smaller fault threshold of $< N_c/4$ must be assumed. On the positive side, we show that there are constructions (e.g. the TSS in [41], or the "trivial" TSS consisting of N_c standard signatures) suitable for CS that respect the $< N_c/3$ threshold for Confirmable Persistence.

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 both Persistence and Confirmable Liveness for $< N_c/3$ corrupted IC peers. The new BB system is secure against (i) a general computationally bounded Byzantine adversary, (ii) in a partially synchronous setting (cf. [22]), where the message delivery delay and the synchronization loss among the entities' clocks are bounded, but the bounds need not to be known for the entities' engagement in the BB system.

Summary of Contributions. Our contributions are as follows:

- Introduction of the first complete framework for the formal study of e-voting BBs captured by the properties of Confirmable Liveness and (Confirmable) Persistence.
- Analysis of the CS BB system [20] 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 [19, Sec. 8].

– A suitably modified variant of the CS protocol that restores Confirmable Liveness and we prove secure 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 assuming honest majority between AB peers, while (ii) Confirmable Liveness holds in the partially synchronous model.

2 Related work

The BB functionality as component of e-voting systems. In a wide range of state-of-the-art e-voting systems (e.g., [31,15,2,14,10,42,32]), the BB is a centralized single point of failure for security analysis. Dini [21] proposed a distributed e-voting service based on [23], 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 [38,40,29,33], a study of requirements [28] or being applicable only to the kiosk-voting based setting [7]. D-DEMOS [16] is a distributed internet-voting system which adopts [32] to the distributed setting. The security in [16] 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 [20] is a reference point for our work, and will be analyzed in Section 5.

Robust public transaction ledgers. In a recently emerged line of research, transaction ledgers used in blockchain applications have been studied under formal cryptographic models. Garay *et al.* [24] formalize *robust public transaction ledgers (RPTLs)* that satisfy (i) *Persistence*, i.e., honest parties agree on their report of transactions published in the ledger in an append-only manner and (ii) *Liveness*, i.e., all transactions in the local view of the honest parties will be eventually reported. Moreover, they prove that Nakamoto's Bitcoin blockchain protocol [35] is an RPTL in a synchronous setting. Taking a step further, Pass *et al.* [36] prove the robustness of Bitcoin blockchain protocol, even when the network is semi-synchronous, i.e., the adversary can impose bounded delays. Very recently, Garay *et al.* [25] introduce a dynamic setting that reflects most real-world scenarios where the number of involved parties may vary over time, while Badertscher *et al.* [4] prove the security of Bitcoin as an RPTL in the universal composability framework [12]. A responsive consensus protocol is one that achieves "instant" confirmation of transactions (i.e., dependent only on network delay), and it is shown in [37], that such protocols exist only assuming a 1/3 bound on the adversary. As a side observation, our protocol matches this bound and this level of responsiveness in the permissioned setting (while [37] is in the permissionless setting) and can be seen as a responsive permissioned distributed ledger. We note that the permissionless setting is not immediately relevant to e-voting, since in e-voting, we need to explicitly permission write access so that it is consistent with the e-voting semantics (e.g., the one voter - one vote principle).

3 Preliminaries

We use κ as the security parameter. We write $f(\kappa) = \mathsf{negl}(\kappa)$ if function f is negligible in κ, and PPT for probabilistic polynomial-time. 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 the following PPT algorithms: (i) the *key generation algorithm* KGen generates a keypair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$ for a secret signing key sk and a public verification key pk; (ii) the *signing algorithm* Sig returns, on input a message m and a signing key sk, a signature $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(m)$; (iii) the *verification algorithm* Vf returns, given pk, a message m and signature σ, a bit $b \leftarrow \mathsf{Vf}_{\mathsf{pk}}(m, \sigma)$.

The correctness of DS requires that for each $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\kappa)$ and a valid message m, it must hold that $\mathsf{Vf}_{\mathsf{pk}}(m, \mathsf{Sig}_{\mathsf{sk}}(m)) = 1$. The security of DS is formalized via the notion of

existential unforgeability against chosen message attacks (EUFCMA, [27]). For a more detailed description of EUFCMA security, cf. Appendix A.1.

Threshold signature schemes. Let $t_s < N$ be two positive integers and $P_1, \ldots, P_N$ a set of peers. A (non-interactive) *threshold signature scheme (TSS)* $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{Combine}, \mathsf{TVf})$ is a tuple of the following five efficient algorithms: (i) the *distributed key generation algorithm* $\mathsf{DistKeygen}(1^\kappa, t_s, N)$ generates a keypair $(\mathsf{tsk}_i, \mathsf{pk}_i)$ for each peer P_i and a public key pk, such that exactly $t_s + 1$ secret keys tsk_i are required to recover the secret key tsk corresponding to pk. The public output is pk together with a tuple of verification keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$; (ii) the *signing algorithm* $\mathsf{ShareSig}_{\mathsf{tsk}_i}(m)$ returns a signature share σ_i of the message m; (iii) the *share verification algorithm* $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, m, (i, \sigma_i))$ outputs 1 iff σ_i is the valid ith signature share of m; (iv) the *share combining algorithm* $\mathsf{Combine}(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, m, (i, \sigma_i)_{i \in S})$, given a subset of $t_s + 1$ valid signature shares on m, outputs a full signature $\sigma \leftarrow \mathsf{TSign}(\mathsf{tsk}, m)$ on m; (v) the *verification algorithm* $\mathsf{TVf}_{\mathsf{pk}}(m, \sigma)$ outputs 0 or 1, depending on whether σ is a valid signature of m.

The *correctness* of TSS requires that for a vector $(\mathsf{tsk}, \mathsf{pk}, \mathsf{tsk}_1, \ldots, \mathsf{tsk}_N, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ 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 *existentially* (t_s, N)*-unforgeable against chosen-message attacks* ((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. We write $\mathsf{Adv}^{\mathsf{tss}}_{\mathcal{A}}(\kappa, t_s, N)$ to denote $\mathcal{A}$'s advantage in breaking the (t_s, N)-EUFCMA-security of TSS. For a more detailed description of (t_s, N)-EUFCMA security, cf. Appendix A.2.

TSS is said to be (t_s, N)-*robust*, if $\mathcal{A}$ controlling t_s peers, cannot prevent honest peers from creating a valid signature. Robustness can only be achieved for $t_s < N/2$, see Gennaro *et al.* [26].

4 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 [20] and the modeling approach of [16] and [24], we formalize BB security via the notions of *(Confirmable) Persistence* and *Confirmable Liveness*.

4.1 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. At the preparation period, 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 not some public data (e.g., public encryption key).
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 [30], if a voter U submitted multiple votes then only the last one 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 the 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 [5]. 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 AB consistency.

4.2 Introducing our security framework

In [20], Culnane and Schneider propose a list of four properties that a secure BB must satisfy:

(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 four properties into a formal security framework. At a high level, our framework conflates the formal approach in distributed e-voting security of Chondros *et al.* [16] with the notion of a *robust public transaction ledger (RPTL)* proposed by Garay *et al.* [24]. 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 [24] are informally expressed as follows:

- *Persistence*: once an honest peer reports an item x as posted, then all honest peers will either (i) agree on the AB position that x should be published, or (ii) not report x.
- *θ-Liveness*: honest peers will report honestly submitted items within some delay bound θ.

Introducing Confirmable θ-Liveness. In the e-voting scenario, we require that the honest users will eventually get a receipt when engaging at the **Posting** protocol. Consequently, we introduce the *Confirmable θ-Liveness* property, suggesting that *any honest user that submits an item x, will obtain a valid receipt for x within time θ, and x will be published on the AB.*

The conflict between Confirmable Liveness and property (bb.3). An important observation is that Confirmable Liveness and (bb.3) cannot be satisfied concurrently if we assume that honest users may submit post requests for clashing items (e.g., in Estonian e-voting [30], as a coercion countermeasure, voters may vote multiple times and only the last counts). To resolve this conflict, we make the following relaxation: we do not require that (bb.3) holds, by

[5] 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).

allowing every successfully posted item to be published. Then, we specify the subset of valid published items for each user via the selection function $\mathsf{Select}(\cdot)$ over the set of published items; the description of $\mathsf{Select}(\cdot)$ depends on the election setting. Note that in the special case where we restrict the honest users from submitting clashing items, we can match (bb.3) by fixing $\mathsf{Select}(\cdot)$ to be the identity function.

Introducing (Confirmable) Persistence. Given the above, to encompass (bb.1) and (bb.4), we introduce the notion of *Persistence* where conflict resolution is achieved by applying $\mathsf{Select}(\cdot)$ on the AB view. Persistence as defined in [24] captures the "unremovability" property (bb.4). To also capture (bb.1), we require that in a setting where t_c out-of N_c IC peers are corrupted, an item x posted by U will not be reported as valid, if at least $t_c + 1$ honest IC peers did not record x at some **Posting** protocol execution. Note that if we require a threshold of less than $t_c + 1$ honest IC peers to be unaware of item x, then (bb.1) becomes unrealistic, as in most distributed systems, an adversary that corrupts t_c peers can block t_c honest peers, and run the execution jointly with the rest $N_c - 2t_c$ honest peers.

Furthermore, we extend Persistence by taking into account a 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 [16], 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:

- The event $\mathsf{Init}(X)$: $\mathsf{Clock}[X] \leftarrow \mathsf{Clock}$, that initializes X by synchronizing its internal clock with the global clock.
- The event $\mathsf{Inc}(\mathsf{Clock}[X])$: $\mathsf{Clock}[X] \leftarrow \mathsf{Clock}[X]+1$, that causes some 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. Any values $\delta, \Delta \in [0, \infty)$ refer to a *semi-synchronous* model, if δ, Δ are known to the system's entities, or a *partially synchronous* model, if δ, Δ are unknown.

Notation. We use κ as the security parameter and 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 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 peers that the adversary may statically corrupt out of the N_c (resp. N_w) total peers of the IC (resp. AB) subsystem. 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 $M_{j,T} \in \mathbb{N}$.

4.3 (Confirmable) Persistence definition

We define Persistence via a security game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\mathsf{Prst}}(1^\kappa, \mathbf{E})$ 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. The game is described in detail in Fig. 1. Intuitively, the goal of $\mathcal{A}$ is to successfully attack the (bb.1) property (winning condition (P.1)) or the (bb.4) property (winning condition (P.2)).

Threat model for Persistence.

(P.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.

(P.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*).

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

(P.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. If the interaction is completed successfully, $\mathcal{C}$ obtains a receipt $\mathsf{rec}[x]$ for x.

Game winning conditions.

The game outputs 1 iff exists an AB peer $AB_j \notin L_{\mathsf{corr}}$ s.t. at least one of the following conditions holds:

(P.1). There is an item x, $t_c + 1$ honest IC peers $\{P_{i_k}\}_{k \in [t_c+1]}$, 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) $x \notin L_{\mathsf{pub},j,T'}$.

Fig. 1. The BB Persistence security game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\mathsf{Prst}}(1^\kappa, \mathbf{E})$, between $\mathcal{C}$ and $\mathcal{A}$.

Game winning conditions.

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

(P.1). There is an item x, $t_c + 1$ honest IC peers $\{P_{i_k}\}_{k \in [t_c+1]}$, 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) $x \notin L_{\mathsf{pub},j,T''}$.

Fig. 2. Game winning conditions for the BB Confirmable Persistence security game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$.

Subsequently, 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 $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ that follows the same steps as $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\mathsf{Prst}}(1^\kappa, \mathbf{E})$, for the special case $t_w = N_w$, except the following differences in the winning conditions for $\mathcal{A}$, that are presented in detail in Fig. 2: (1) for every $k \in [N_w]$, the published data on AB_k should always verify successfully, and (2) the inconsistent AB_j referred in either of (P.1) or (P.2) conditions may be any (malicious) AB peer.

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*

Threat model for Confirmable Liveness. The threat model satisfies conditions (P.I)-(P.IV) in Fig. 1.

Protocol execution under the presence of $\mathcal{A}$. The initialization by the challenger $\mathcal{C}$ and the engagement of $\mathcal{A}$ and $\mathcal{C}$ in the **Setup** phase, the **Posting** protocol, and the **Publishing** protocol is as in Fig. 1.

Game winning conditions.

The game outputs 1 iff exists an honest user U, an item x and a moment T s.t. the following holds:

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

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

(L.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 s.t. $\mathsf{VerifyRec}(z, x, \mathsf{cr}_U, \mathsf{params}) = \mathsf{accept}$.

(b) There is a party $AB_j \notin L_{\mathsf{corr}}$, s.t. for any moment T_j, exists a moment $T'_j \geq T_j$ s.t. $x \notin L_{\mathsf{pub},j,T'_j}$.

Fig. 3. The BB Liveness security game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\theta-\mathsf{C.Live}}(1^\kappa, \mathbf{E})$ between $\mathcal{C}$ and $\mathcal{A}$.

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*

$$\mathbf{Pr}\left[\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\mathsf{Prst}}(1^\kappa, \mathbf{E}) = 1\right] = \mathsf{negl}(\kappa).$$

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*

$$\mathbf{Pr}\left[\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E}) = 1\right] = \mathsf{negl}(\kappa).$$

4.4 Confirmable θ-Liveness definition

We define θ-Confirmable Liveness via a security game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\theta-\mathsf{C.Live}}(1^\kappa, \mathbf{E})$ 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 manages to prevent 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. 3.

Definition 2 (Confirmable θ-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 with* N_c *IC peers and* N_w *AB peers. We say that* **BB** achieves Confirmable θ-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*

$$\Pr\left[\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\theta-\mathsf{C.Live}}(1^\kappa, \mathbf{E}) = 1\right] = \mathsf{negl}(\kappa).$$

5 The Culnane-Schneider (CS) BB system

In this section, we provide an outline of CS BB system as presented in [20] adopted in our terminology, and analyze its security guarantees and weaknesses under the framework introduced in Sec. 4. 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 [20]), i.e., $N_w = 1$. The fault-tolerance threshold on the number of corrupted IC peers, t_c, that CS requires is

$$t_c < N_c/3 \quad \text{and} \quad t_s + 1 = N_c - t_c \tag{1}$$

5.1 Description 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 $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{TVf}, \mathsf{Combine})$ (cf. Sec. 3). Then, each IC peer P_i runs $\mathsf{KGen}(1^\kappa)$ to obtain a signing key sk_i and a verification key vk_i, while all IC peers jointly execute $\mathsf{DistKeygen}(1^\kappa, t_s, N_c)$ to produce a secret keys $(\mathsf{tsk}_1, \ldots, \mathsf{tsk}_{N_c})$, implicitly defining tsk, and the corresponding public output $\mathsf{pk}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c})$.

Upon key generation, the IC peers broadcast $\boldsymbol{pk} := \{\mathsf{pk}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c}), (\mathsf{vk}_1, \ldots, \mathsf{vk}_{N_c})\}$ to all other entities. The public parameters params include the description of $\mathsf{DS}, \mathsf{TSS}, \mathcal{P}$, as well as $\boldsymbol{pk}$. Finally, every user U engages with SA in an interaction to obtain her private input cr_U. As mentioned in Sec. 4.1, cr_U could be a singing key, or an authentication password.

From a timing aspect, the CS BB system runs in consecutive periods. Namely, 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 $\mathsf{VerifyRec}(\mathsf{rec}[x], x, \mathsf{cr}_U, \mathsf{params}) := \mathsf{TVf}_{\mathsf{pk}}((p, x), \mathsf{TSign}(\mathsf{tsk}, (p, x)))$. The **Posting** protocol is presented in more detail in Fig. 6 of Appendix B.1.

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 specified 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* [34, Section 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. The **Publishing** protocol is presented in more detail in Fig. 7 of Appendix B.1.

5.2 Attacking the liveness of the CS BB system

As informally argued in [19, Sec. 8] (the full version of [20]), 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.

In this section, we demonstrate an attack against the Confirmable Liveness of CS under our framework. Although our attack falls outside the threat model of [20], 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 cleverly 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 points out 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 [20] to be properly justified. On the other hand, as we describe in Section 6 and prove in Section 7, 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 [20], and consists of seven steps described below. The steps are illustrated in Fig. 8 of Appendix B.2 for the simple case where $N_c = 4$ and $t_c = 1$:

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 $\mathcal{R}_h^c$. On the other hand, the 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 [20] (expressed via condition

(L.3) in Fig. 3), 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.

The strength of our liveness attack. The attack described above requires the collaboration of malicious users and corrupted IC peers. Nonetheless, it is important to note that during the **Publishing** protocol, the t_c malicious peers behave consistently towards the $N_c - 2t_c$ peers in $\mathcal{H}_{\mathsf{in}}$ by providing the same records and signature databases. In addition, the fact that they send no information to the peers in $\mathcal{H}_{\mathsf{out}}$ is a malicious behavior that the peers *cannot prove* to the peers in $\mathcal{H}_{\mathsf{in}}$, i.e., the peers in $\mathcal{H}_{\mathsf{in}}$ cannot tell whether the peers in $\mathcal{H}_{\mathsf{out}}$ are (i) honest and they did not receive anything, or (ii) malicious and claim some false denial of service. As a result, the peers in $\mathcal{H}_{\mathsf{in}}$ can not distinguish whether (i) the t_c malicious peers, or (ii) the t_c peers in $\mathcal{H}_{\mathsf{out}}$, are the ones that deviate from the protocol guidelines. This shows that the attacker can deviate from the guidelines and still not be detected, within the context of the CS IC consensus.

5.3 TSS Fault-Tolerance Requirements for Confirmable Persistence

In [20] 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. Gennaro et al. show in [26] that TSS can achieve robustness only if $t_s < N_c/2$. This contradicts the CS requirements in Eq. (1). 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. However, this issue seems to be less worrisome than the liveness attack in the previous section. Usually in TSS literature it is assumed that $t_s = t_c$. In [41], Shoup proposes a more general definition of $(k, t_s, N_c) - \mathsf{TSS}$ (with a construction) where t_s is the number of malicious peers and $k \geq t_s + 1$ is the number of TSS shares needed to combine a valid signature. This would allow us to still get robustness by having $t_s < N_c/2$, but we can also set $k = N_c - t_c$, as CS requires in Eq. (1). Shoup's construction uses trusted setup, but if one prefers to avoid this, then they may use the trivial TSS defined in Sec. A.3.

In conclusion, CS BB could still achieve the bound $t_c < N_c/3$, but care must be taken when choosing TSS. Instantiating CS BB with an arbitrary TSS, might not give the expected security guarantees. More thorough treatment of this issue, together with the attack and the proof discussed above, is provided in Appendix B.3.

6 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 model, and Persistence in asynchronous model, against a general Byzantine adversary, assuming a threshold of $t_c < N_c/3$ corrupted IC peers. Our novel **Publishing** protocol requires (i) eventual message delivery and (ii) the existence of a *Binary Consensus (BC) protocol* BC with $t_c < N_c/3$ Byzantine fault tolerance. Both these conditions are met in the partially synchronous model (cf. [22]). Recall that a BC protocol satisfies: 1) *Termination*: all honest peers will decide on some value, 2) *Agreement*: all honest peers decide on the same value, and 3) *Validity*: if all honest peers engage in the protocol with input $b \in \{0, 1\}$, then they all decide on b.

The protocol consists of the following phases:

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

(i). Its *direct view* of local records, denoted by $\mathsf{View}_{i,p} := \langle \tilde{B}_{i,1,p}, \ldots, \tilde{B}_{i,N_c,p} \rangle$: by setting $\tilde{B}_{i,j,p} \leftarrow \perp$, 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 $\{?, 0, 1\}$ and expresses *the opinion of P_i on the validity of P_j's behavior*. Initially, only $b_{i,i}$ 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 some value 1/0 for all $j \in [N_c]$ (which we prove it happens, cf. Lemma 3), it engages in the **Consensus** phase, that we describe shortly.

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

■ The **Collection** phase: upon initialization, P_i signs its local record $B_{i,p}$, followed by a tag denoted by RECORD, and broadcasts $\big((\text{RECORD}, B_{i,p}), \mathsf{Sig}_{\mathsf{sk}_i}(\text{RECORD}, B_{i,p})\big)$ to all IC peers. During this phase, 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 verifies the validity of the message by checking 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 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 and engages in the **Consensus** phase for the record of P_k (marked as malicious). 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 in the first place.

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 $\tilde{B}^i_{k,j,p} \leftarrow R_{k,j,p}$. Then, P_i updates $\mathsf{View}_{i,j,p}$ according to the cases below:

> **(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 message for the record of P_j[6].

– *When P_i has fixed $N_c - t_c$ opinion bits*, it 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 lack of direct view of P_k for P_j's record until it received P_i's request.

For every P_j that P_i has broadcast a request $\big((\text{REQUEST_VIEW}, j), \mathsf{Sig}_{\mathsf{sk}_i}(\text{REQUEST_VIEW}, j)\big)$, P_i waits until it collects $N_c - t_c - 1$ distinct signed responses that verify successfully. During this

[6] 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.

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 $\tilde{B}_{i,j,p}$ and fixes $b_{i,j}$ as follows: (a) if $\tilde{B}_{i,j,p} \neq \perp$, all responses for non-$\perp$ records are at least t_c and match $\tilde{B}_{i,j,p}$, then P_i fixes $b_{i,j}$ to 1; (b) if $\tilde{B}_{i,j,p} = \perp$, all responses for non-$\perp$ records are at least $t_c + 1$ and 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; (c) else, it sets $\tilde{B}_{i,j,p} \leftarrow \perp$ and fixes $b_{i,j}$ to 0[7].

In any case, upon fixing $b_{i,j}$, P_i ignores any message for the record of P_j. Whenever P_i fixes $b_{i,j}$ to some value 1/0, the view of P_i for P_j's record reaches a "binary" state (cf. Lemma 3), i.e., P_i's opinion on P_j's record is that it is either (i) a fixed non-$\perp$ record, or (ii) set to $\perp$. Thus, honest peers can reach consensus for all records, as described below.

■ The **Consensus** phase: when P_i fixes its "opinion" $b_{i,j}$ to some value 1/0 for P_j for every $j \in [N_c]$, it engages in the *partially synchronous Binary Consensus (BC) protocol* BC, for the question "*Is my view of P_j's record set to a non-$\perp$ value?*".

Namely, each peer P_i engages in BC with input $b_{i,j} = 1$, if $\tilde{B}_{i,j,p} \leftarrow \tilde{B}^i_{j,p} \neq \perp$ and $b_{i,j} = 0$, if $\tilde{B}_{i,j,p} \leftarrow \perp$, as set at the **Collection** phase. The BC properties guarantee termination and agreement, i.e. all honest peers agree on the same value. Moreover, during **Collection** phase, all $N_c - t_c \geq t_c$ honest peers have received each others' (fixed) records, so they engage in BC for an honest peer P_j on input 1. Hence, by the validity property of BC, the honest peers will agree on 1 for P_j. After all N_c BC executions have terminated (the IC peers have decided), for every $j \in [N_c] \setminus \{i\}$, P_i updates its view as follows: If $\tilde{B}_{i,j,p} \neq \perp$, but P_i decided 0 for P_j, then it updates as $\tilde{B}_{i,j,p} \leftarrow \perp$.

The peer P_i considers the **Consensus** phase finished, only when it has completed the BC protocols for all peers' records. Then, they proceed to the **Finalization** phase described below.

■ The **Finalization** phase: having updated its direct view $\mathsf{View}_{i,p} := \langle \tilde{B}_{i,1,p}, \ldots, \tilde{B}_{i,N_c,p} \rangle$ for every pair $(p, x) \in \bigcup_{j:\tilde{B}_{i,j,p} \neq \perp} \tilde{B}_{i,j,p}$, peer P_i defines the set $N_{i,p}(x)$ that denotes the number of IC peers that, according to P_i's view, have included (p, x) in their records. Formally, we write $N_{i,p}(x) := \big|\{j \in [N_c] : (p, x) \in \tilde{B}_{i,j,p}\}\big|$. 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[8].

As shown in Lemma 3, 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 proceed to the **Publication** phase described below.

■ The **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, we write $\mathsf{ShareSig}(\mathsf{tsk}_i, (p, B_{i,p})) := \bigcup_{(p,x) \in B_{i,p}} \mathsf{ShareSig}(\mathsf{tsk}_i, (p, x))$. Then, it sends to AB the message $\big((p, B_{i,p}), \mathsf{ShareSig}(\mathsf{tsk}_i, (p, B_{i,p}))\big)$.

In turn, the AB receives and records threshold signature shares for posted items. For every item (p, x) that AB receives $N_c - t_c$ distinct successfully verified signatures shares $(j, \sigma_j)_{j \in S}$, where S is a subset of $\geq N_c - t_c$ IC peers, it adds (p, x) to the record B_p (initialized as empty) and computes a TSS signature on (p, x) as $\mathsf{TSign}(\mathsf{tsk}, (p, x)) \leftarrow \mathsf{Combine}\big(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c}, (p, x), (j, \sigma_j)_{j \in S}\big)$.

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

[8] 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 (cf. Theorem 2). 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.

At the end of period p, AB fixes the signed record $\mathsf{TSign}(\mathsf{tsk}, (p, B_p)) := \bigcup_{(p,x)\in B_p} \mathsf{TSign}(\mathsf{tsk}, (p, x))$ and publishes the record

$$\mathsf{ABreceipt}[p, B_p] := \big((p, B_p), \mathsf{TSign}(\mathsf{tsk}, (p, B_p))\big) .$$

Let $\mathsf{Prec}[p]$ be the set of all periods preceding p. The total view of AB at some moment T during period p, denoted by $L_{\mathsf{pub},T}$ is the union of the agreed and published BB records for all periods in $\mathsf{Prec}[p]$. Let $\mathsf{VerifyPub}(L_{\mathsf{pub},T}, \mathsf{params}) := \bigwedge_{\tilde{p}\in\mathsf{Prec}[p]} \bigwedge_{(\tilde{p},x)\in B_{\tilde{p}}} \mathsf{TVf}\big(\mathsf{pk}, (\tilde{p}, x), \mathsf{TSign}(\mathsf{tsk}, (\tilde{p}, x))\big)$.

7 Properties of the New BB System

In this section, we prove the properties of the BB system described in Sec. 6. 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. Recall that since we have only one AB peer, we denote the view of the AB as $L_{\mathsf{pub},T} = L_{\mathsf{pub},1,T}$.

In our setting, we assume that, given the running time of the protocol steps, the message delivery delay δ and the synchronization loss bound Δ are small enough with respect to the intervals $[T_{\mathsf{begin},p}, T_{\mathsf{barrier},p}], [T_{\mathsf{barrier},p}, T_{\mathsf{end},p}]$ that specify phase length and changing in each period p. We stress that this restriction does not violate partial synchronicity, as δ, Δ need not to be known to the IC peers for executing the protocol.

Persistence. The Persistence of the new BB system holds in the asynchronous model, preserving the security of the BB protocols per period under sequential composition due to period advancement. This holds because the view of the AB in some moment T at period p, is the union of the views of all periods preceding p which is invariant to the publishing order of each view. The proof of the following theorem is given in Appendix C.1.

Theorem 1. *Let $N_c, t_c, t_s \in \mathbb{N}$, where $t_c < N_c/3$ and $t_s \geq N_c - t_c - 1$, and let $\delta = \Delta = \infty$. Let* TSS *be a (t_s, N_c)-EUFCMA-secure TSS. Then, the BB system described in Sec. 6 with N_c IC peers over* TSS *achieves Persistence for tolerance thresholds $(t_c, 0)$.*

Remark 1 (From Persistence to Confirmable Peristence). The new BB system does not support Confirmable Persistence because although a malicious AB could not attack condition (P.1) of Fig. 2 (this part of the proof of Theorem 1 is valid even against a malicious AB), it could breach condition (P.2) by discarding all signature shares on items published at earlier periods. A way to address this problem with a simple modification would be to replace a single AB with a subsystem of N_w AB peers. Then, the IC peers would broadcast their local records to all peers at the **Publication** phase and the AB peers would proceed as the centralized AB described in Section 6. In addition, reading of the AB data would be via honest majority (i.e. $t_w < N_w/2$) of matching data on the AB peers, as in [16]. This way, any alteration of previous published data by some AB peer would be detected by an auditor, while the fact that the majority of the AB peers would be consistent, would also hold this peer accountable. Thus, a minority of malicious AB peers could not attack (P.2) - and still not (P.1) of course - so Confirmable Persistence would be achievable.

Confirmable Liveness. We prove the Confirmable Liveness of the new BB system in the partially synchronous model against Byzantine adversaries. The proof is given in Appendix C.2.

Theorem 2 (Confirmable Liveness). *Let $N_c, t_c, t_s \in \mathbb{N}$ such that (a) $t_c < N_c/3$ and (b) $t_c \leq t_s < N_c - t_c$, and $\delta, \Delta \in \mathbb{R}_{\geq 0}$. Let* DS *be an EUFCMA-secure signature scheme and* TSS *be a (t_s, N_c)-EUFCMA-secure TSS. Let* BC *be a BC protocol with t_c-out-of-N_c fault tolerance, that is partially synchronous for delay message bound δ and synchronization loss bound Δ. Then the BB system described in Sec. 6 with N_c IC peers over* DS, TSS, *and* BC achieves θ-Confirmable Liveness *for fault tolerance thresholds $(t_c, 0)$, delay message bound δ and synchronization loss bound Δ, and for every $\theta \geq \Delta + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}$.*

References

1. Michel Abdalla, Sara K. Miner, and Chanathip Namprempre. Forward-Secure Threshold Signature Schemes. In *CT-RSA 2001*, pages 441–456, 2001.
2. Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX 2008*, pages 335–348, 2008.
3. Yonatan Aumann and Yehuda Lindell. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. *J. Cryptology*, 23(2):281–343, 2010.
4. Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a Transaction Ledger: A Composable Treatment. pages 324–356, 2017.
5. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure Computation Without Authentication. In *CRYPTO 2005*, pages 361–377, 2005.
6. Josh Benaloh. Verifiable secret-ballot Elections. PhD thesis, Yale University, 1987.
7. Josh Benaloh, Michael D. Byrne, Bryce Eakin, Philip T. Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. In *EVT/WOTE 2013*, 2013.
8. Jose Beuchat. Realization of a Secure Distributed Bulletin Board. Bern University of Applied Sciences, 2012. Master's thesis.
9. Alexandra Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC 2003*, pages 31–46, 2003.
10. Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Vanessa Teague, Roland Wen, Zhe Xia, and Sriramkrishnan Srinivasan. Using Prêt à Voter in Victoria State Elections. In *EVT/WOTE 2012*, 2012.
11. Craig Burton, Chris Culnane, and Steve Schneider. vVote: Verifiable Electronic Voting in Practice. *IEEE Security & Privacy*, 14(4):64–73, 2016.
12. Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS 2001*, pages 136–145, 2001.
13. David Chaum. SureVote: Technical Overview. In *WOTE 2001*, 2001.
14. David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *Security & Privacy, IEEE*, 6(3):40–46, 2008.
15. David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A Practical Voter-Verifiable Election Scheme. In *ESORICS 2005*, pages 118–139, 2005.
16. Nikos Chondros, Bingsheng Zhang, Thomas Zacharias, Panos Diamantopoulos, Stathis Maneas, Christos Patsonakis, Alex Delis, Aggelos Kiayias, and Mema Roussopoulos. D-DEMOS: A Distributed, End-to-End Verifiable, Internet Voting System. In *ICDCS 2016*, pages 711–720, 2016.
17. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *EUROCRYPT 1997*, pages 103–118, 1997.
18. Chris Culnane, Peter Y. A. Ryan, Steve A. Schneider, and Vanessa Teague. vvote: A verifiable voting system. *ACM Trans. Inf. Syst. Secur.*, 18(1):3:1–3:30, 2015.
19. Chris Culnane and Steve Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. *CoRR*, abs/1401.4151, 2014.
20. Chris Culnane and Steve A. Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *CSF 2014*, pages 169–183, 2014.
21. Gianluca Dini. A secure and available electronic voting service for a large-scale distributed system. *Future Generation Computer Systems*, 19(1):69–85, 2003.
22. Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the Presence of Partial Synchrony. *J. ACM*, 35(2):288–323, April 1988.
23. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT 1992*, pages 244–251, 1992.
24. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In *EUROCRYPT 2015*, pages 281–310, 2015.
25. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In *CRYPTO 2017*, pages 291–323, 2017.
26. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and Efficient Sharing of RSA Functions. In *CRYPTO 1996*, pages 157–172, London, UK, UK, 1996. Springer-Verlag.
27. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
28. Severin Hauser and Rolf Haenni. A Generic Interface for the Public Bulletin Board Used in UniVote. In *CeDEM 2016*, pages 49–56, 2016.
29. James Heather and David Lundin. The Append-Only Web Bulletin Board. In *FAST 2008*, pages 242–256, 2008.
30. Sven Heiberg and Jan Willemson. Verifiable internet voting in estonia. In *EVOTE 2014*, pages 1–8, 2014.
31. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES 2005*, pages 61–70, 2005.

32. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-End Verifiable Elections in the Standard Model. In *EUROCRYPT 2015*, pages 468–498, 2015.
33. Roland Krummenacher. Implementation of a Web Bulletin Board for E-Voting Applications. *MSE Seminar on E-Voting. Institute for Internet Technologies and Applications*, 2010.
34. Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
35. Satoshi Nakamoto. Bitcoin: a peer-to-peer electronic cash system, 2008.
36. Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the Blockchain Protocol in Asynchronous Networks. In *EUROCRYPT 2017*, pages 643–673, 2017.
37. Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.
38. R. A. Peters. A Secure Bulletin Board. Eindhoven University of Technology, 2005. Master's thesis.
39. Michael K. Reiter. The Rampart Toolkit for Building High-integrity Services. In *TPDS 1995*, pages 99–110, 1995.
40. Daniel Sandler and Dan S. Wallach. Casting votes in the auditorium. In *EVT*, 2007.
41. Victor Shoup. Practical Threshold Signatures. In *EUROCRYPT 2009*, pages 207–220, 2009.
42. Filip Zagórski, Richard T Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *ACNS 2013*, 2013.

A Security definitions of signature schemes and threshold signature schemes

A.1 EUFCMA security of signature schemes

Le $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ be a signature scheme. We provide the definition of EUFCMA security game $\mathcal{G}^{\mathcal{A}}_{\mathsf{DS}}(1^\kappa)$ in Fig. 4.

The EUFCMA security game $\mathcal{G}^{\mathcal{A}}_{\mathsf{DS}}(1^\kappa)$.

1. The challenger $\mathcal{C}_{\mathsf{DS}}$ runs KGen to generate a keypair $(\mathsf{pk}, \mathsf{sk})$. The adversary $\mathcal{A}$ receives pk as input.
2. The adversary may ask the $\mathcal{C}_{\mathsf{DS}}$ to sign a number of messages. To query the i-th signature, $\mathcal{A}$ submits a message m_i to $\mathcal{C}_{\mathsf{DS}}$, which returns $\sigma_i \leftarrow \mathsf{Sig}(\mathsf{sk}, m)$.
3. $\mathcal{A}$ outputs a message m^* and a signature σ^*.

The game outputs 1 ($\mathcal{A}$ wins) iff (i) $\mathsf{Vf}(\mathsf{pk}, m^*, \sigma^*) = 1$, and (ii) for every i, it holds that $(m^*, \sigma^*) \neq (m_i, \sigma_i)$.

Fig. 4. The EUFCMA security game $\mathcal{G}^{\mathcal{A}}_{\mathsf{DS}}(1^\kappa)$ between the challenger $\mathcal{C}_{\mathsf{DS}}$ and the adversary $\mathcal{A}$.

Definition 3 (Security signature schemes). *Let* $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *be a signature scheme. We say that* DS *is* EUFCMA-secure *if for every PPT adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathcal{G}^{\mathcal{A}}_{\mathsf{DS}}(1^\kappa) = 1] = \mathsf{negl}(\kappa) \ .$$

A.2 (t_s, N)-EUFCMA security of threshold signature schemes

Let $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{TVf}, \mathsf{Combine})$ be a threshold signature scheme. We provide the definition of (t_s, N)-EUFCMA security game $\mathcal{G}^{\mathcal{A}}_{\mathsf{TSS}}(1^\kappa, t_s, N)$ in Fig. 5.

Definition 4 (Security of threshold signature schemes). *Let t_s, N be values polynomial in the security parameter κ. Let* $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{TVf}, \mathsf{Combine})$ *be a threshold signature scheme. We say that* TSS *is* (t_s, N)-EUFCMA-secure *if for every PPT adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathcal{G}^{\mathcal{A}}_{\mathsf{TSS}}(1^\kappa, t_s, N) = 1] = \mathsf{negl}(\kappa) \ .$$

We define $\mathcal{A}$'s advantage $\mathsf{Adv}^{\mathrm{tss}}_{\mathcal{A}}(\kappa, t_s, N)$ as the probability that $\mathcal{A}$ wins the game $\mathcal{G}^{\mathcal{A}}_{\mathsf{TSS}}(1^\kappa, t_s, N)$, taken over all coin tosses, i.e.

$$\mathsf{Adv}^{\mathrm{tss}}_{\mathcal{A}}(\kappa, t_s, N) := [\Pr[\mathcal{G}^{\mathcal{A}}_{\mathsf{TSS}}(1^\kappa, t_s, N) = 1] \ .$$

The (t_s, N)-EUFCMA security game $\mathcal{G}^{\mathcal{A}}_{\mathsf{TSS}}(1^\kappa, t_s, N)$.

1. First, $\mathcal{A}$ decides on the set $L_{\mathsf{corr}} \subset \{1, \ldots, N\}$ of corrupted players, and sends it to the challenger $\mathcal{C}_{\mathsf{TSS}}$. $\mathcal{A}$ is allowed to act on behalf of each $P_i \in L_{\mathsf{corr}}$. Then $\mathsf{DistKeygen}(1^\kappa, t_s, N)$ is executed. During this, $\mathcal{C}_{\mathsf{TSS}}$ plays the role of all honest players P_i. At the end of this step, $\mathcal{A}$ gets access to pk.
2. $\mathcal{A}$ can make up to polynomial number of signing queries as follows: for $i \in \{1, \ldots, N\} \setminus L_{\mathsf{corr}}$, $\mathcal{A}$ submits pair (i, m) to $\mathcal{C}_{\mathsf{TSS}}$. Then, $\mathcal{C}_{\mathsf{TSS}}$ returns $\sigma_i \leftarrow \mathsf{ShareSig}(\mathsf{tsk}_i, m)$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a message m^* and a signature σ^*. Let $\mathcal{V} = L_{\mathsf{corr}} \cup \mathcal{S}$, where $\mathcal{S}$ is the subset of players for which $\mathcal{A}$ made a signing query (i, m^*).

The game outputs 1 ($\mathcal{A}$ wins) iff (i) $\mathsf{TVf}(\mathsf{pk}, m^*, \sigma^*) = 1$, and (ii) $|\mathcal{V}| \leq t_s$.

Fig. 5. The (t_s, N)-EUFCMA security game $\mathcal{G}^{\mathcal{A}}_{\mathsf{TSS}}(1^\kappa, t_s, N)$ between the challenger $\mathcal{C}_{\mathsf{TSS}}$ and the adversary $\mathcal{A}$.

The Posting Protocol.

$U \to P_i$**:** User U broadcasts item x with credential cr_U to all peers P_i, $i \in [N_c]$.

$P_i \to P_j$**:** Upon receiving (x, cr_U) from U, each IC peer P_i checks that (i) cr_U is valid for U (i.e $(U, x) \in \mathsf{Accept}$), and (ii) x does not clash with any previously posted item x' during p or previous periods. If both checks are successful, then it broadcasts $\big((p, x, \mathsf{cr}_U), \mathsf{Sig}_{\mathsf{sk}_i}(p, x, \mathsf{cr}_U)\big)$ to all other IC peers.

P_i**:** P_i waits for messages $\big((p, x, \mathsf{cr}_U), \mathsf{Sig}_{\mathsf{sk}_j}(p, x, \mathsf{cr}_U)\big)$ from peers P_j, $j \neq i$, and appends them to the dataset of received signed items $D_{i,p}$. Upon receiving $N_c - t_c$ valid signatures on (p, x, cr_U) (including its own), P_i adds (p, x) to its local BB record $B_{i,p}$ for period p.

$P_i \to U$**:** P_i sends its TSS share $((p, x), \mathsf{ShareSig}_{\mathsf{tsk}_i}(p, x))$ to U.

U**:** The user U waits for valid signatures $\big((p, x), \mathsf{ShareSig}_{\mathsf{tsk}_j}(p, x)\big)$ from $N_c - t_c \geq t_s + 1$ peers P_j. When this happens, let S_U be a set of $t_s + 1$ peers from which U can combine the share. The receipt for x that U obtains is set as

$$\mathsf{rec}[x] := \mathsf{TSign}_{\mathsf{tsk}}(p, x) \leftarrow \mathsf{Combine}\big(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c}, (p, x), (j, \sigma_j)_{j \in S_U}\big) \quad .$$

Fig. 6. The **Posting** protocol of the CS BB system during period p w.r.t. the posting policy $\mathcal{P} = \big(\mathsf{Accept}, \mathsf{Select}(\cdot)\big)$, for fault tolerance threshold of t_c out-of N_c IC peers $P_1, \ldots, P_{N_c}$, and a (t_s, N_c)-TSS. Selection function $\mathsf{Select}(\cdot)$ is the identity function.

A.3 Trivial TSS

In our work, we refer to the following *trivial* (t_s, N)*-TSS*, where $t_s < N$ can be any non-negative integer: given an EUFCMA-secure signature scheme DS, each party P_i has an independently generated public and secret key. The signature share of P_i on m is equal to her signature on m and signature shares are combined by concatenation. The TSS verification on $(\sigma_1, \ldots, \sigma_t)$ succeeds only if each individual signature verification succeeds and $t > t_s$. Clearly, trivial TSS is (t_s, N)-EUFCMA-secure. We note that the trivial TSS slightly deviates from the previous definition of TSS, as there is no secret key tsk shared between the parties.

B Appendix of Section 5

B.1 The Posting and Publishing protocols of the CS BB system

We provide detailed descriptions of the **Posting** and **Publishing** protocols of the CS BB system in Fig. 6 and Fig. 7, respectively.

B.2 Illustrating the attack against the liveness of the CS BB system

We illustrate the steps of our liveness attack in Fig. 8 for the case where $N_c = 4$ and $t_c = 1$.

The Optimistic Protocol.

$P_i \to P_j$**:** Each P_i broadcasts $\big((p, B_{i,p}), \mathsf{Sig}_{\mathsf{sk}_i}(p, B_{i,p})\big)$ to all P_j.

– Each P_i waits until it receives $N_c - t_c$ valid signatures $\big((p, B_{j,p}), \rho_j := \mathsf{Sig}_{\mathsf{sk}_j}(p, B_{j,p})\big)$ from different IC peers (including its own). Let I be the set of those peers.

– If $\sharp\big\{P_j \in I : B_{j,p} = B_{i,p} \wedge \mathsf{Vf}_{\mathsf{pk}_j}(\rho_j) = 1\big\} < N_c - t_c$, then P_i broadcasts a message $\big((p, D_{i,p}), \mathsf{Sig}_{\mathsf{sk}_i}(p, D_{i,p})\big)$ to all IC peers, indicating that it engages in the Fallback protocol (see below).

$P_i \to$ AB**:** Each P_i sends $\big((p, B_{i,p}), \mathsf{ShareSig}_{\mathsf{tsk}_i}(p, B_{i,p})\big)$ to AB.

AB**:** Waits for messages $\big((p, B_{j,p}), \sigma_j = \mathsf{ShareSig}_{\mathsf{tsk}_j}(p, B_{j,p})\big)$ from at least $N_c - t_c \geq t_s + 1$ peers P_j.

– Let S be a set of those $\geq N_c - t_c$ peers that agree on the contents of the BB and let B_p be the board of agreed-on contents. Compute $\mathsf{TSign}_{\mathsf{tsk}}(p, B_p) \leftarrow \mathsf{Combine}\big(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c}, (p, B_p), (j, \sigma_j)_{j \in S}\big)$.

– Publish $\mathsf{ABreceipt}[p, B_p] := \big((p, B_p), \mathsf{TSign}_{\mathsf{tsk}}(p, B_p)\big)$.

The Fallback Protocol.

$P_i \to P_j$**:** Each P_i broadcasts $\big((p, D_{i,p}), \sigma_i = \mathsf{Sig}_{\mathsf{sk}_i}(p, D_{i,p})\big)$ to all P_j.

P_i**:** Each P_i updates its database with the new data, that is, with the signatures it is missing, and then updates its board. More precisely, if $\mathsf{Vf}_{\mathsf{pk}_j}((p, D_{j,p}), \sigma_j) = 1$, then P_i will insert all new signatures in $D_{j,p}$ to its database $D_{i,p}$ (i.e., P_i sets $D_{i,p} \leftarrow D_{i,p} \cup D_{j,p}$). For any x, if P_i has $N_c - t_c$ valid signatures on (p, x, cr_U) then it adds (p, x) to $B_{i,p}$.

– P_i broadcasts a message $\big((p, B_{i,p}), \mathsf{Sig}_{\mathsf{sk}_i}(p, B_{i,p})\big)$ indicating that it re-engages in Step 1 of the Optimistic protocol (see above) for the updated record $B_{i,p}$.

Fig. 7. The **Publishing** protocol of the CS BB system during period p, for fault tolerance threshold of t_c out-of N_c IC peers $P_1, \ldots, P_{N_c}$, and a (t_s, N_c)-TSS.

B.3 Confirmable Persistence of CS BB

In the **Posting** protocol of the CS BB system, if the centralized AB is honest, then it will neither accept inconsistently signed data nor remove any existing published items. As a result, CS satisfies the properties (bb.1) and (bb.4), which are captured in our Persistence definition.

As for Confirmable Persistence, all data published by the AB must be signed by a TSS signature. Thus, even if AB is malicious, then, clearly, any attack that includes illegitimate item posting or published item removal will be detected as long as the t_s out-of N_c TSS fault tolerance threshold is preserved, since the malicious AB can not forge signatures on inconsistent records without the consent of at least a number of honest peers. This raises the following conflict: TSS fault tolerance must satisfy the restrictions in equation Eq. (1), namely that $t_s + 1 > 2N_c/3$, however, the robustness requirement $t_s < N_c/2$ stated in [26] must also hold, otherwise liveness is trivially breached. These two requirements are clearly contradictory.

Since the robustness is necessary, we study the security of CS in the case of $t_s < N_c/2$. The following paragraph argues that for $t_s < N_c/2$ and $t_c \geq N_c/4$, there is an attack against confirmable persistence. Specifically, the attack achieves winning condition (P.1) in Fig. 2 (property (bb.1)). Subsequently, we show that for $t_c < N_c/4$ CS does achieve confirmable persistence.

Description of the Confirmable Persistence attack. Assume that $N_c \geq 4$, $N_c/4 \leq t_c < N_c/3$ and a (t_s, N_c)-EUFCMA-secure TSS, where $t_c \leq t_s < N_c/2$. Our attack consists of the three steps described below and illustrated in Fig. 9, for the case of $N_c = 10$, $t_c = 3$ and $t_s = 4$:

Step 1: Let $t_h = t_s - t_c$. Assume that in period p a user U posts an item (p, x) but does not obtain a receipt. More precisely, assume that the adversary blocked the communication so that at the end of the posting protocol, $(p, x) \in B_{i,p}$ for only one honest peer P_i.

Step 2: During the Optimistic protocol, the adversary blocks messages so that all $t_c = t_s - t_h$ corrupted peers and exactly $(t_s + 1) - t_c = t_h + 1$ honest peers (thus, in total $t_s + 1$ peers) threshold sign a record B_p such that $(p, x) \notin B_p$ and send it to the AB. More precisely, only

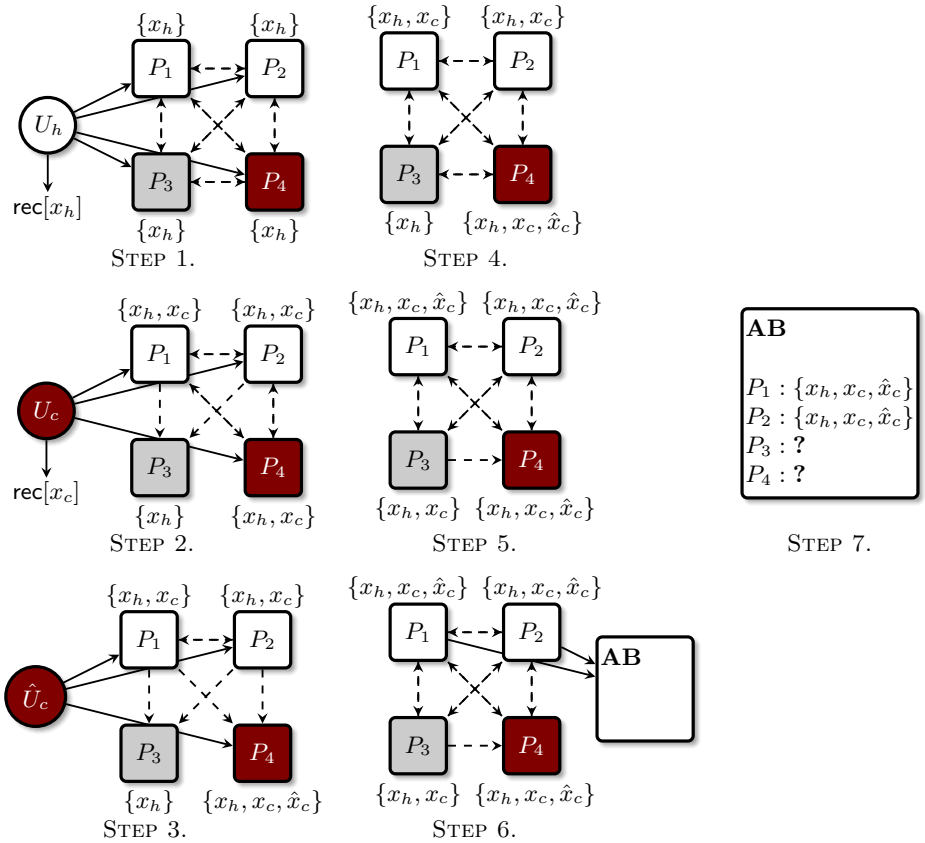


Fig. 8. Attacking the liveness of CS for $N_c = 4$ and $t_c = 1$, where $\mathcal{H}_{\mathsf{in}} = \{P_1, P_2\}$ and $\mathcal{H}_{\mathsf{out}} = \{P_3\}$. The malicious entities are colored in red. The elements in $\{\cdot\}$ denote the local records of each IC peer of publishable items for period p at the end of the given step.

the said $t_s + 1$ peers will obtain $N_c - t_c$ signatures on B_p. Thus, a malicious AB can construct a receipt on B_p.

Step 3: The remaining $N_c - t_c - (t_h + 1)$ honest peers together with t_c corrupted peers execute the Fallback and the Optimistic protocol, and agree on a B'_p s.t. $(x, p) \in B'_p$. Since $N_c \geq 4$ and $N_c/4 \leq t_c \leq t_s < N_c/2$, we have that

$$N_c \geq 2t_s - t_c + 2 \Rightarrow N_c - (t_h + 1) \geq t_s + 1 \ .$$

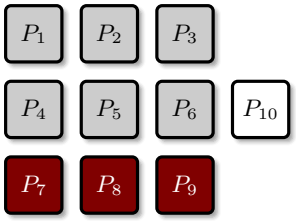
Thus, a malicious AB can construct a receipt on B'_p. By the above, a malicious AB can output two valid threshold signatures for a period p, i.e.,

$$\begin{aligned} w &= \mathsf{ABreceipt}[p, B_p] = \big((p, B_p), \sigma = \mathsf{TSign}_{\mathsf{tsk}}((p, B_p))\big) \wedge \\ w' &= \mathsf{ABreceipt}[p, B'_p] = \big((p, B'_p), \sigma' = \mathsf{TSign}_{\mathsf{tsk}}((p, B'_p))\big) \\ &\wedge w \neq w' \wedge \mathsf{TVf}(\mathsf{pk}, (p, B_p), \sigma) = \mathsf{TVf}(\mathsf{pk}, (p, B'_p), \sigma') = 1, \end{aligned}$$

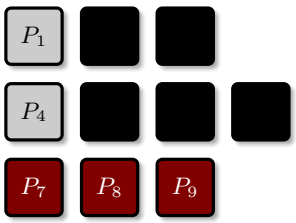
hence, Confirmable Persistence is violated.

The extent of the Confirmable Persistence attack. We show that even with $t_s < N_c/2$, CS BB can still tolerate a threshold of $t_c < N_c/4$ malicious peers. More generally, if we forget about the robustness bound (which is only for liveness), then CS BB has Confirmable Persistence for $t_c < (t_s + 1)/2$.

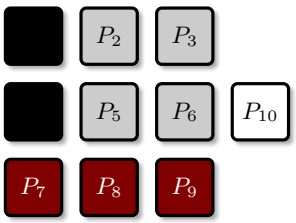
To prove this claim, we make use of Lemma 1 showing that if an adversary $\mathcal{A}$ wins the Confirmable Persistence game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$, then for every period p, at least $t_h + 1$ honest peers have contributed their signature share for the TSS of the published record for p.



STEP 1. During the **Posting** protocol, P_{10} is the only honest peer that received the post request for x. The honest peers $P_1, \ldots, P_6$ colored in gray are unaware of x.



STEP 2. During the Optimistic protocol only the $t_s + 1 = 5$ peers P_1, P_4, P_7, P_8, P_9 obtain $N_c - t_c = 7$ signatures and send their TSS shares for $B_{i,p} \not\ni (x, p)$ to the malicious AB.



STEP 3. The $N_c - t_c + 1 = 8$ peers $P_2, P_3, P_5, \ldots, P_{10}$ start Fallback and Optimistic protocols, and after that send their TSS shares on $B'_{ip} \ni (x, p)$ to the malicious AB.

Fig. 9. Attack against Confirmable Persistence for $N_c = 10$, $t_c = 3$ and $t_s = 4$. The corrupted nodes P_7, P_8, P_9 are colored in red.

Lemma 1. *Let $N_c, t_c, t'_c, t_s \in \mathbb{N}$ such that $t_c \leq t_s$, $t'_c \leq t_c$, and let $\delta = \Delta = \infty$. Let $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{TVf}, \mathsf{Combine})$ be a (t_s, N_c)-EUFCMA-secure TSS. Let $\mathcal{A}$ be an arbitrary PPT adversary that corrupts t'_c IC peers and $t_h := t_s - t'_c$. If $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E}) = 1$ for the CS BB system then for any period p, with probability $1 - \mathsf{negl}(\kappa)$ there is a set $\mathcal{H} := \{P_{i_k}\}_{k \in [t_h+1]}$ of honest IC peers that output $((p, B_p), \mathsf{ShareSig}(\mathsf{tsk}_{i_k}, (p, B_p)))$ for $k \in [t_h + 1]$, where B_p is the published record for p.*

Proof. Let $T > T_{\mathsf{end},p}$ be a moment after the end of period p. If $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E}) = 1$ then $\mathsf{VerifyPub}(L_{\mathsf{pub},T}) = \mathsf{accept}$, therefore $\mathsf{ABreceipt}[p, B_p] = ((p, B_p), \sigma = \mathsf{TSign}(\mathsf{tsk}, (p, B_p))) \in L_{\mathsf{pub},T}$ such that $\mathsf{TVf}(\mathsf{pk}, (p, B_p), \sigma) = 1$. Let $\mathcal{H} = \{P_{i_k}\}$ be the set of honest IC peers that output $\mathsf{ShareSig}(\mathsf{tsk}_{i_k}, (p, B_p))$.

Suppose to the contrary that less than $t_h + 1$ honest peers output a threshold signature on (p, B_p). We construct the following adversary $\mathcal{A}_{\mathsf{TSS}}$ that breaks the static (t_s, N_c)-EUFCMA security of TSS by emulating the game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ playing the role of the challenger. The security reduction is executed as shown below:

1. After $\mathcal{A}$ responds with the set of corrupted IC peers L_{corr}, $\mathcal{A}_{\mathsf{TSS}}$ in turn sends L_{corr} to the challenger $\mathcal{C}_{\mathsf{TSS}}$ of the (t_s, N_c)-EUFCMA security game and hence corrupts the same subset of peers.
2. $\mathcal{A}_{\mathsf{TSS}}$ engages in the **Setup** phase with $\mathcal{A}$ playing the role of SA and the honest IC peers as follows:
   (a) $\mathcal{A}_{\mathsf{TSS}}$ specifies the posting policy $\mathcal{P} = (\mathsf{Accept}, \mathsf{Select}(\cdot))$ and a signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$. It sends the description of $\mathcal{P}, \mathsf{DS}$ and TSS to $\mathcal{A}$.

(b) $\mathcal{A}_{\mathsf{TSS}}$ engages with $\mathcal{A}$ in the joint execution of $\mathsf{DistKeygen}(1^\kappa, t_s, N_c)$ by the IC peers, by forwarding the messages of $\mathcal{A}$ during the interaction of itself and $\mathcal{C}$ in the joint execution at the beginning of the (t_s, N_c)-EUFCMA security game. At the end of the joint execution, both $\mathcal{A}_{\mathsf{TSS}}$ and $\mathcal{A}$ obtain the TSS public key pk.

(c) On behalf of each honest peer P_i, $\mathcal{A}_{\mathsf{TSS}}$ runs $\mathsf{KGen}(1^\kappa)$ to obtain a signing key sk_i and a verification key vk_i. It sends the generated verification keys to $\mathcal{A}$ which provides $\mathcal{A}_{\mathsf{TSS}}$ with the malicious peers' verification keys (if this does not happen, then $\mathcal{A}_{\mathsf{TSS}}$ aborts).

(d) When interacting with a corrupted user U controlled by $\mathcal{A}$ for the computation of private input s_U, $\mathcal{A}_{\mathsf{TSS}}$ acts normally on behalf of SA.

3. Whenever an honest peer P_i engaging in the emulation of **Posting** and **Publishing** protocol executions has to provide a TSS share signature on some message (i, m), then $\mathcal{A}_{\mathsf{TSS}}$ makes the query (i, m) to $\mathcal{C}_{\mathsf{TSS}}$ and obtains the response $\sigma_i \leftarrow \mathsf{ShareSig}(\mathsf{tsk}_i, m)$. Then, $\mathcal{A}_{\mathsf{TSS}}$ uses σ_i in the emulation step.
4. After $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ is completed, if for the aforementioned $T > T_{\mathsf{end,p}}$, $\mathcal{A}$ (playing the role of a malicious AB) has returned to $\mathcal{A}_{\mathsf{TSS}}$ a $\mathsf{ABreceipt}[p, B_p] = ((p, B_p), \sigma) \in L_{\mathsf{pub},T}$ such that $\mathsf{TVf}(\mathsf{pk}, (p, B_p), \sigma) = 1$, then $\mathcal{A}_{\mathsf{TSS}}$ returns $((p, B_p), \sigma)$ to $\mathcal{C}_{\mathsf{TSS}}$.

Since $t_c \leq t_s$, if $\mathcal{A}$ wins $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$, then it has corrupted no more than t_s IC peers. In addition, $\mathcal{A}$ makes no singing queries for (p, B_p) during its engagement with the BB protocols, which means that, by assumption, $\mathcal{A}$ obtains less than $t_h + 1$ additional threshold signatures, i.e. the ones provided by the honest peers. In total, $\mathcal{A}$ obtains no more than $t'_c + t_h \leq t_s$ threshold signatures, so the restrictions of the (t_s, N_c)-EUFCMA security game (cf. Fig. 5) are preserved. Therefore, $\mathsf{TVf}(\mathsf{pk}, (p, B_p), \sigma) = 1$ implies that $((p, B_p), \sigma)$ is a successful forgery against the static (t_s, N_c)-EUFCMA security of TSS. Thus, if $\mathcal{A}$ wins $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ with non-negligible probability α, then $\mathcal{A}_{\mathsf{TSS}}$ outputs a successful forgery with the same probability α, which contradicts to the (t_s, N_c)-EUFCMA-security of TSS. We conclude that, with probability $1 - \mathsf{negl}(\kappa)$, there exists a set $\mathcal{H}$ of $t_h + 1$ honest peers that output $((p, B_p), \sigma)$ for $k \in [t_h + 1]$. □

Given Lemma 1, we can prove the following theorem.

Theorem 3. *Let $N_c, t_c, t_s \in \mathbb{N}$, such that $t_s \leq N_c$ and let $\delta = \Delta = \infty$. Let $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ be an EUFCMA-secure signature scheme and $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{TVf}, \mathsf{Combine})$ be a (t_s, N_c)-EUFCMA-secure TSS. For all $t_c < (t_s + 1)/2$ the CS BB system, with N_c IC peers over DS and TSS achieves Confirmable Persistence for tolerance threshold $(t_c, 1)$.*

Proof. Let $\mathcal{A}$ be an any PPT adversary against the Confirmable Persistence game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ that corrupts $t'_c \leq t_c$ IC peers. As before $t_h := t_s - t'_c$. Note that condition $t_s \geq t_c$ of Lem. 1 is satisfied since $2t_c < t_s + 1$. We show that with probability $1 - \mathsf{negl}(\kappa)$ neither of the properties (P.1) nor (P.2) can hold if $\mathsf{VerifyPub}(L_{\mathsf{pub},j,T}, \mathsf{params}) = \mathsf{accept}$ for every moment T.

There is a moment T' such that if $\mathsf{VerifyPub}(L_{\mathsf{pub},T'}, \mathsf{params}) = \mathsf{accept}$, then (P.1) cannot hold: Suppose to the contrary that there exists an item (p, x), honest peers $\mathcal{H}' = \{P_{i_k}\}_{k \in [t_c+1]}$, and moments T', T'', such that
(i) $T' \leq T''$,
(ii) $(x, p) \in L_{\mathsf{pub},T'}$, and
(iii) $(x, p) \notin L_{\mathsf{post},i_k,T''}$, for any $k \in [t_c + 1]$.

Since $\mathsf{VerifyPub}(L_{\mathsf{pub},T'}) = \mathsf{accept}$ and TSS is (t_s, N_c)-unforgeable, it follows from Lem. 1 that for some period p at least $t_h + 1$ honest peers signed B_p such that $(p, x) \in B_p$ no later than T'. Recall that an honest peer threshold signs B_p if it has obtained $N_c - t_c$ signatures on (p, B_p).

First, we look at the case where less than $N_c - t_c - t'_c$ of the signatures are created by honest peers. We show that this allows us to construct an efficient adversary $\mathcal{A}_s$ that breaks EUFCMA security of DS. Let $\mathcal{C}_{\mathsf{DS}}$ be the challenger of EUFCMA game (cf. Fig. 3). The construction of $\mathcal{A}_s$ is as follows.

1. Challenger $\mathcal{C}_{\mathsf{DS}}$ generates a keypair $(\mathsf{pk}, \mathsf{sk})$ and sends pk to $\mathcal{A}_s$.
2. Adversary $\mathcal{A}_s$ runs $\mathcal{A}$ that returns L_{corr} and in particular the set of corrupted IC peers $\mathcal{IC}$ where $|\mathcal{IC}| = t'_c$.
3. $\mathcal{A}_s$ assigns the key pk to a random honest peer $P_i \notin \mathcal{IC}$ and sets $\mathcal{S} \leftarrow \emptyset$.
4. $\mathcal{A}_s$ runs **Setup** phase as usual and interacts with $\mathcal{A}$ exactly as the challenger of the Confirmable Persistence game would (in particular, it interacts for the honest entities), only with the following exceptions:
   (a) If $\mathcal{A}_s$ needs to sign a message under pk, then it asks it from $\mathcal{C}_{\mathsf{DS}}$ ($\mathcal{A}_s$ itself does not know sk).
   (b) Whenever an honest peer receives a valid signature ρ of a message (p, B_p) under a public key pk' of some IC peer, it adds ρ to $\mathcal{S}$ if $\mathcal{S}$ has no signatures under pk' before.
5. Finally, if P_i's signature σ is in $\mathcal{S}$ and P_i is the honest peer that did not sign (p, B_p), then $\mathcal{A}_s$ returns σ as a forgery.

We know that $|\mathcal{S}| \geq N_c - t_c$ and hence at least $N_c - t_c - t'_c$ of the signatures in $\mathcal{S}$ are valid signatures of distinct honest peers. We assumed that less than $N_c - t_c - t'_c$ of honest peers output a signature on (p, B_p), therefore $\mathcal{A}$ has forged a signature of at least one honest peer. Since public keys of all honest peers come from the same distribution and $\mathcal{A}_s$ assigned pk to a random honest peer, then at least with the probability $\frac{1}{N_c - t'_c}$, $\mathcal{A}$ forged a signature for pk. Therefore $\mathcal{A}_s$ wins the EUFCMA game with a non-negligible probability in κ.

In the second case, we assume that at least $N_c - t_c - t'_c$ of the signatures are from honest peers. For these peers $(p, x) \in B_{i,p}$. However, the honest peers only sign one local record so these $N_c - t_c - t'_c$ peers are all different from the $t_c + 1$ honest peers that have not included x in their list of posted items at moment $T'' \geq T'$ hence, later than p. Since $(N_c - t_c - t'_c) + (t_c + 1) > N_c - t_c$, we have a contradiction with condition (ii).

There is a moment T' such that if $\mathsf{VerifyPub}(L_{\mathsf{pub},T'}, \mathsf{params}) = \mathsf{accept}$, then (P.2) cannot hold: First, let us show that for a single period p with probability $1 - \mathsf{negl}(\kappa)$, adversary $\mathcal{A}$ can output a valid threshold signature for only one database.

Assume to the contrary that $\mathcal{A}$ outputs

$$\begin{aligned} w =& \mathsf{ABreceipt}[p, B_p] = (p, B_p, \sigma = \mathsf{TSign}_{\mathsf{tsk}}(p, B_p)) \ , \\ w' =& \mathsf{ABreceipt}[p, B'_p] = (p, B'_p, \sigma' = \mathsf{TSign}_{\mathsf{tsk}}(p, B'_p)) \end{aligned}$$

for $B_p \neq B'_p$, such that $\mathsf{Vf}_{\mathsf{pk}}((p, B_p), \sigma) = \mathsf{Vf}_{\mathsf{pk}}((p, B'_p), \sigma') = 1$.

From Lem. 1, we get that the set $\mathcal{H}$ of $t_h + 1$ honest peers threshold-signed (p, B_p) and the set $\mathcal{H}'$ of $t_h + 1$ honest peers threshold-signed (p, B'_p). According to the protocol description, $\mathcal{H} \cap \mathcal{H}' = \emptyset$.

An honest peer threshold-signs B_p if it has obtained $N_c - t_c$ signatures on (p, B_p). Suppose now that less than $N_c - t_c - t'_c$ of the signatures are from honest peers. In that case we can construct an adversary $\mathcal{A}_s$ that breaks unforgeability of Σ, the construction of $\mathcal{A}_s$ is the same as above.

Therefore, at least $N_c - t_c - t'_c$ of the signatures are from honest peers, denoted by the set $\mathcal{S}$. Since $t_h + 1 > t_c$, it holds that

$$|\mathcal{S}| + |\mathcal{H}'| \geq N_c - 2t_c + t_h + 1 > N_c - t_c \ ,$$

so at least one honest peer P_i threshold signs B'_p and sends signature on B_p. If P_i first threshold-signs B'_p, then the protocol is over for P_i, and it would not sign B_p. So, it must first sign B_p and then threshold sign the database B'_p, which implies $B_p \subseteq B'_p$.

Arguing similarly, we get that there must me be some honest peer P_j that first signed B'_p and later threshold signed B_p, which implies that $B'_p \subseteq B_p$. Therefore, $B_p = B'_p$ which contradicts our assumption.

Now we show that (P.2) can not hold. We have that $L_{\mathsf{pub},T'} = \bigcup_{\tilde{p} \in \mathsf{Prec}[p']} \big(\mathsf{ABreceipt}[\tilde{p}, B_{\tilde{p}}]\big)$, where p' is the period number at moment T', and $\mathsf{VerifyPub}\big(L_{\mathsf{pub},T'}\big) = \mathsf{accept}$. Suppose there

exists a moment $T'' \geq T'$ and an item (p, x) such that $(p, x) \in L_{\mathsf{pub},T'}$ and $(p, x) \notin L_{\mathsf{pub},T''}$. As $(p, x) \in L_{\mathsf{pub},T'}$, it follows that there exists $p \leq p'$ such that $(p, x) \in B_p$. On the contrary, as $(p, x) \notin L_{\mathsf{pub},T''}$ and T'' is later than p, we have that there is another published record $\hat{B}_p$ s.t. $(p, x) \notin \hat{B}_p$. Since for every period p adversary can output a valid signature for exactly one record, we have a contradiction. □

Discussion. Theorem 3 implies that

1). If $t_s < N_c/2$ (robustness bound), then we can show the bound $t_c < N_c/4 + 1/2$. Together with the attack in beginning of this Section, we get that $t_c < N_c/4$ and this bound is tight.

2). If $t_s = 2N_c/3$ (original CS bound), then $t_c < N_c/3$ which is exactly the bound that Culnane and Schneider claim.

Moreover, the attack described in the beginning of this Section depends on the bound $t_s < N_c/2$. We argue that there are simple (but perhaps not ideal) ways to bypass $t_s < N_c/2$ bound and hence possible to avoid Confirmable Persistence attack. However, it can be seen as a warning that a TSS scheme can not be utilized as a black-box tool in order to realize CS BB system.

Reason for the robustness bound $t_c < N_c/2$, is that typically TSS is studied in the case when $t_c = t_s$. Hence, if $t_s \geq N_c/2$, then signing is only possible with the help from malicious peers and robustness cannot be achieved because they can block the signature creation.

In [41] Shoup proposes a more general $(k, t_s, N_c) - \mathsf{TSS}$ construction where t_s is the number of malicious peers and $k \geq t_s + 1$ is the number of signature shares needed to combine a valid signature. This scheme also assumes that $t_s < N_c/2$ but k can be larger than $t_s + 1$. Such definition seems to be more appropriate for the CS BB setting since we may take $k = \lfloor 2N_c/3 \rfloor + 1$, $t_s = \lceil N_c/3 \rceil - 1$ and still maintain robustness. Besides [41], we are aware of only [1] that proposes a $(k, t_s, N_c) - \mathsf{TSS}$. Both schemes however use a trusted dealer which preferably should be avoided in the e-voting scenario.

Another possibility is to use the trivial (k, t_s, N_c)-TSS scheme described in Sec. A.3. Trivial scheme has robustness for any t_s such that $t_s \leq N_c - t_s - 1$, since key generation is done locally and there are enough honest peers to produce a valid signature. Of course, this TSS construction loses the main advantage of TSS, which is to achieve lower communication and verification complexity than sending and verifying N_c signatures individually.

We do not claim that it is impossible to construct a (k, t_s, N_c)-TSS which would avoid the issues mentioned above. However, care must be taken when instantiating CS BB, since most TSS schemes in the literature are not proved secure as $(k, t, N_c) - \mathsf{TSS}$. Setting the parameter t_s greater than $N_c/2$ in case of $(t_s, N_c) - \mathsf{TSS}$ might not be secure even if $t_c < N_c/2$.

C Appendix for Section 7

C.1 Proof of Theorem 1

First, we prove the following Lemma stating that if an adversary $\mathcal{A}$ wins the Confirmable Persistence game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$, then for every period p and $(p, x) \in B_p$ that is published by the AB, at least $t_h + 1$ honest peers have contributed their TSS signature share for (p, x). Note that since it holds for Confirmable Persistence game it also holds for the Persistence game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,t_w}_{\mathsf{Prst}}(1^\kappa, \mathbf{E})$.

Lemma 2. *Let $N_c, t_c, t'_c, t_s \in \mathbb{N}$ such that $t_c \leq t_s$, $t'_c \leq t_c$, and let $\delta = \Delta = \infty$. Let $\mathsf{TSS} = (\mathsf{DistKeygen}, \mathsf{ShareSig}, \mathsf{ShareVerify}, \mathsf{TVf}, \mathsf{Combine})$ be a (t_s, N_c)-EUFCMA-secure TSS. Let $\mathcal{A}$ be an arbitrary PPT adversary that corrupts t'_c IC peers and $t_h := t_s - t'_c$. If $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E}) = 1$ for the BB system described in Sec. 6, then for any period p, $T > T_{\mathsf{end},p}$ and $(p, x) \in L_{\mathsf{pub},T}$, with probability $1 - \mathsf{negl}(\kappa)$ there is a set $\mathcal{H} := \{P_{i_k}\}_{k \in [t_h+1]}$ of honest IC peers that provide AB with $\mathsf{ShareSig}(\mathsf{tsk}_{i_k}, (p, x))$ for $k \in [t_h + 1]$.*

Proof. Let $T > T_{\mathsf{end},p}$ be a moment after the end of period p. If $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E}) = 1$ then $\mathsf{VerifyPub}\big(L_{\mathsf{pub},T}\big) = \mathsf{accept}$, therefore $\mathsf{ABreceipt}[p, B_p] = ((p, B_p), \sigma = \mathsf{TSign}(\mathsf{tsk}, (p, B_p))) \in L_{\mathsf{pub},T}$ such that $\bigwedge_{(p,x')\in B_p} \mathsf{TVf}\big(\mathsf{pk}, \big(p, x'\big), \mathsf{TSign}(\mathsf{tsk}, (p, x'))\big) = \mathsf{accept}$. Let $\mathcal{H} = \{P_{i_k}\}$ be the set of honest IC peers that output $\mathsf{ShareSig}(\mathsf{tsk}_{i_k}, (p, x))$, where $(p, x) \in B_p$.

Suppose to the contrary that less than $t_h + 1$ honest peers output a threshold signature on (p, x). We construct the following adversary $\mathcal{A}_{\mathsf{TSS}}$ that breaks the static (t_s, N_c)-EUFCMA security of TSS by emulating the game $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ playing the role of the challenger. The security reduction is executed as shown below:

1. After $\mathcal{A}$ responds with the set of corrupted IC peers L_{corr}, $\mathcal{A}_{\mathsf{TSS}}$ in turn sends L_{corr} to the challenger $\mathcal{C}_{\mathsf{TSS}}$ of the (t_s, N_c)-EUFCMA security game and hence corrupts the same subset of peers.
2. $\mathcal{A}_{\mathsf{TSS}}$ engages in the **Setup** phase with $\mathcal{A}$ playing the role of SA and the honest IC peers as follows:
   (a) $\mathcal{A}_{\mathsf{TSS}}$ specifies the posting policy $\mathcal{P} = \big(\mathsf{Accept}, \mathsf{Select}(\cdot)\big)$ and a signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$. It sends the description of $\mathcal{P}, \mathsf{DS}$ and TSS to $\mathcal{A}$.
   (b) $\mathcal{A}_{\mathsf{TSS}}$ engages with $\mathcal{A}$ in the joint execution of $\mathsf{DistKeygen}(1^\kappa, t_s, N_c)$ by the IC peers, by forwarding the messages of $\mathcal{A}$ during the interaction of itself and $\mathcal{C}$ in the joint execution at the beginning of the (t_s, N_c)-EUFCMA security game. At the end of the joint execution, both $\mathcal{A}_{\mathsf{TSS}}$ and $\mathcal{A}$ obtain the TSS public key pk.
   (c) On behalf of each honest peer P_i, $\mathcal{A}_{\mathsf{TSS}}$ runs $\mathsf{KGen}(1^\kappa)$ to obtain a signing key sk_i and a verification key vk_i. It sends the generated verification keys to $\mathcal{A}$ which provides $\mathcal{A}_{\mathsf{TSS}}$ with the malicious peers' verification keys (if this does not happen, then $\mathcal{A}_{\mathsf{TSS}}$ aborts).
   (d) When interacting with a corrupted user U controlled by $\mathcal{A}$ for the computation of private input s_U, $\mathcal{A}_{\mathsf{TSS}}$ acts normally on behalf of SA.
3. Whenever an honest peer P_i engaging in the emulation of **Posting** and **Publishing** protocol executions has to provide a TSS share signature on some message (i, m), then $\mathcal{A}_{\mathsf{TSS}}$ makes the query (i, m) to $\mathcal{C}_{\mathsf{TSS}}$ and obtains the response $\sigma_i \leftarrow \mathsf{ShareSig}(\mathsf{tsk}_i, m)$. Then, $\mathcal{A}_{\mathsf{TSS}}$ uses σ_i in the emulation step.
4. After $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ is completed, if for the aforementioned $T > T_{\mathsf{end,p}}$, $\mathcal{A}$ (playing the role of a malicious AB) has returned to $\mathcal{A}_{\mathsf{TSS}}$ a $\mathsf{ABreceipt}[p, B_p] = ((p, B_p), \sigma) \in L_{\mathsf{pub},T}$ such that $\mathsf{ABreceipt}[p, B_p] = ((p, B_p), \sigma = \mathsf{TSign}(\mathsf{tsk}, (p, B_p))) \in L_{\mathsf{pub},T}$ such that

$$\bigwedge_{(p,x')\in B_p} \mathsf{TVf}\big(\mathsf{pk}, \big(p, x'\big), \mathsf{TSign}(\mathsf{tsk}, (p, x'))\big) = \mathsf{accept}\ ,$$

   then $\mathcal{A}_{\mathsf{TSS}}$ returns $((p, B_p), \sigma)$ to $\mathcal{C}_{\mathsf{TSS}}$.

Since $t_c \leq t_s$, if $\mathcal{A}$ wins $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$, then it has corrupted no more than t_s IC peers. In addition, $\mathcal{A}$ makes no singing queries for $(p, x) \in B_p$ during its engagement with the BB protocols, which means that, by assumption, $\mathcal{A}$ obtains less than $t_h + 1$ additional threshold signatures, i.e. the ones provided by the honest peers. In total, $\mathcal{A}$ obtains no more than $t'_c + t_h \leq t_s$ threshold signatures, so the restrictions of the (t_s, N_c)-EUFCMA security game (cf. Fig. 5) are preserved. Therefore, $\mathsf{TVf}\big(\mathsf{pk}, \big(p, x\big), \mathsf{TSign}(\mathsf{tsk}, (p, x)\big) = 1$ implies that $((p, x), \sigma)$ is a successful forgery against the static (t_s, N_c)-EUFCMA security of TSS. Thus, if $\mathcal{A}$ wins $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c}_{\mathsf{C.Prst}}(1^\kappa, \mathbf{E})$ with non-negligible probability α, then $\mathcal{A}_{\mathsf{TSS}}$ outputs a successful forgery with the same probability α, which contradicts to the (t_s, N_c)-EUFCMA-security of TSS. We conclude that, with probability $1 - \mathsf{negl}(\kappa)$, there exists a set $\mathcal{H}$ of $t_h + 1$ honest peers that provide AB with $\mathsf{ShareSig}(\mathsf{tsk}_{i_k}, (p, x))$ □

Then, we apply Lemma 2 to prove Theorem 1 below:

Proof. Let $\mathcal{A}$ be an arbitrary PPT adversary against Persistence game $\mathcal{G}_{\mathsf{Prst}}^{\mathcal{A},\delta,\Delta,t_c,t_w}(1^\kappa, \mathbf{E})$ that corrupts $t'_c \leq t_c$ IC peers. We set $t_h := t_s - t'_c$. Note that the condition $t_c \leq t_s$ of Lemma 2 is satisfied since $t_s \geq N_c - t_c - 1 > 2t_c - 1$ and therefore $t_s \geq 2t_c \geq t_c$. We show that, with probability $1 - \mathsf{negl}(\kappa)$, neither (P.1) nor (P.2) from Fig. 1 hold.

(P.1) cannot hold: Note that since AB is honest, then $\mathsf{VerifyPub}(L_{\mathsf{pub},T}, \mathsf{params}) = \mathsf{accept}$ for every moment T. Suppose to the contrary that there exists an item (p, x), honest peers $\mathcal{H}' = \{P_{i_k}\}_{k\in[t_c+1]}$, and moments T', T'', such that
(i) $T' \leq T''$,
(ii) $(p, x) \in L_{\mathsf{pub},T'}$, and
(iii) $(p, x) \notin L_{\mathsf{post},i_k,T''}$, for any $k \in [t_c + 1]$.

As $\mathsf{VerifyPub}(L_{\mathsf{pub},T'}) = \mathsf{accept}$ and TSS is (t_s, N)-EUFCMA-secure, it follows from Lemma 2 that with probability $1 - \mathsf{negl}(\kappa)$ there are at least $t_h + 1$ honest peers that provide AB with $\mathsf{ShareSig}(\mathsf{tsk}_{i_k}, (p, x))$ no later than moment T'. By condition (ii) above, we have that $(t_h + 1) + (t_c + 1) \geq t_s + 2 > N_c - t_c$, so there exists an honest peer that threshold signed an item (p, x) that's not in its database. According to protocol description, an honest peer never does it. Hence, we have a contradiction.

(P.2) cannot hold: This is trivially true for our protocol. Clearly, for all moments $T' < T''$ and all $(p, x) \in L_{\mathsf{pub},j,T'}$, we have $(p, x) \in L_{\mathsf{pub},j,T''}$ since honest AB never removes items. □

C.2 Proof of Theorem 2

IC Consensus in the Publishing protocol. As a first step, we prove that all honest peers will include all honestly posted items for which a valid receipt has been generated.

Lemma 3. *Let $N_c, t_c \in \mathbb{N}$ such that $t_c < N_c/3$ and $\delta, \Delta \in \mathbb{R}_{\geq 0}$. Let DS be an EUFCMA-secure signature scheme and BC be a Binary Consensus (BC) protocol with t_c-out-of-N_c fault tolerance, that is partially synchronous for message delivery delay bound δ and synchronization loss bound Δ. Let (p, x) be an honestly posted item for which the user has obtained a valid receipt. Then, every honest IC peer P_i engaging at the* **Publishing** *protocol over DS and BC, will include (p, x) in $B_{i,p}$ at the end of the* **Finalization** *phase, with $1 - \mathsf{negl}(\kappa)$ probability.*

Proof. By the EUFCMA security of DS, with $1 - \mathsf{negl}(\kappa)$ probability, a malicious peer will not forge a signed message for some honest peer. In the rest of the proof, we assume that all valid signatures under sk_i are indeed generated by pk_i. We prove the lemma via the following claim:

Claim 1 *Let $P_i, P_{i'}$ be two honest IC peers that have engaged in the BC protocol for P_j's record with inputs $b_{i,j}$ and $b_{i',j}$ respectively. Then, the direct views of $P_i, P_{i'}$ for the record of P_j when engaging in the BC protocol are such that if $\tilde{B}_{i,j,p}, \tilde{B}_{i',j,p} \neq \perp$, then $\tilde{B}_{i,j,p}$ and $\tilde{B}_{i',j,p}$ are both equal to some record $\tilde{B}_{j,p}$.*

Proof of Claim 1: Assume that $\tilde{B}_{i,j,p}, \tilde{B}_{i',j,p} \neq \perp$. By the description of the **Collection** phase, whenever an honest peer sets an opinion bit to 0 for some peer, the respective record is set to $\perp$. Thus, it holds that $b_{i,j} = b_{i',j} = 1$. W.l.o.g., assume that $\tilde{B}_{i,j,p} = \tilde{B}_{j,p}$; we will show that $\tilde{B}_{i',j,p} = \tilde{B}_{j,p}$. We recall that updating a non-$\perp$ record to a different non-$\perp$ record is not allowed at **Collection** phase. By protocol description, P_i has set $\tilde{B}_{i,j,p}$ to $\tilde{B}_{j,p}$ due to either one of the following cases:

1. *$P_i = P_j$, i.e. $\tilde{B}_{i,j,p}$ is the record $B_{i,p}$ that P_i has set at the end of the* **Posting** *protocol*: in this case, P'_i will eventually receive the message $\big((\text{RECORD}, B_{i,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, B_{i,p})\big)$ broadcast by P_i at the **Collection** phase. Given that all $N_c - t_c \geq t_c + 1$ honest peers will re-broadcast $B_{i,p}$ and that, except from $\mathsf{negl}(\kappa)$ probability, no malicious peer can forge P_i's signature, P'_i will eventually set $d_{i',i}$ to $t_c + 1$, and by case (C.1), fix $b_{i',i} = 1$ and $\tilde{B}_{i',j,p} = \tilde{B}_{i',i,p} = B_{i,p}$, .

2. *$\tilde{B}_{i,j,p}$ was set to $\tilde{B}_{j,p}$ when P_i received $\big((\text{RECORD}, \tilde{B}_{j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, \tilde{B}_{j,p})\big)$ by peer P_j:* then P_i has re-broadcast this message at the **Collection** phase, in the format $\big(V_{i,j}, \mathsf{Sig}_{\mathsf{sk}_i}(V_{i,j})\big)$, where $V_{i,j} := \big((\text{VIEW}, j), ((\text{RECORD}, \tilde{B}_{i,j,p}), \mathsf{Sig}_{\mathsf{sk}_j}(\text{RECORD}, \tilde{B}_{i,j,p}))\big)$. The latter implies that $P_{i'}$ has eventually received it and set $\tilde{B}^{i'}_{i,j,p} \leftarrow \tilde{B}_{j,p}$. Therefore, since $b_{i',j} \neq 0$, the case (C.2) can not hold, so it must hold that $\tilde{B}_{i',j,p} = \tilde{B}_{i,j,p} = \tilde{B}_{j,p}$.
3. *$\tilde{B}_{i,j,p}$ was set to $\tilde{B}_{j,p}$ by checking case (C.1):* in this case, the variable $d_{i,j}$ has reached the value $t_c + 1$, i.e., there are at least $t_c + 1$ non-$\perp$ records in P_i's view must be equal to $\tilde{B}_{j,p}$. Since there are no more than t_c malicious peers, at least one honest peer P_k has obtained $\tilde{B}_{j,p}$, in order to broadcast it at the **Collection** phase. There are two subcases:
   3.1) If $P_k = P_{i'}$, then since $b_{i',j} \neq 0$, $P_{i'}$ never updated as $\tilde{B}_{i',j,p} \leftarrow \perp$ according to (C.2), so it holds that $\tilde{B}_{i',j,p} = \tilde{B}_{j,p}$.
   3.2) If $P_k \neq P_{i'}$, then both P_i and $P_{i'}$ have eventually received $\tilde{B}_{j,p}$ at **Collection** phase (at least by P_k). Therefore, since $b_{i,j} \neq 0$ and $b_{i',j} \neq 0$, case (C.2) can not hold, so it must hold that $\tilde{B}_{i',j,p} = \tilde{B}_{j,p}$.

(End of Proof of Claim 1) $\dashv$

By Claim 1, we get that all honest peers agree on each others' records before entering in the **Consensus** phase. This implies the following facts:

- For each honest peer P_k, all honest peers will engage in BC on input 1, thus they will decide on 1, by the validity property. The latter implies that if for some peer P_j s.t. $\tilde{B}_{i,j,p} \neq \perp$, but P_i decided 0 for P_j, then it can safely set $\tilde{B}_{i,j,p} \leftarrow \perp$, as P_j must be malicious.
- By the description of the CS **Posting** protocol (cf. Section 5 and Fig. 6, every honestly posted item (p, x) requires $N_c - 2t_c \geq t_c + 1$ honest peers for receipt generation. Therefore, before entering the **Consensus** phase all $N_c - t_c$ honest peers will include (p, x) in their records.

By the above two facts, we have at the **Finalization** phase, the view of every honest peer P_i contains all $N_c - t_c$ honest peers' records which, in turn contain all honestly posted items for which a receipt has been generated. Therefore, when finalizing its record $B_{i,p}$ the following hold:

- For all honestly posted item (p, x) for which a receipt has been generated, it holds that $N_{i,p}(x) \geq N_c - t_c \geq t_c + 1$. Thus, by case (F.1), P_i will add (p, x) in $B_{i,p}$.
- For every item $(p, \hat{x})$ s.t. $N_{i,p}(\hat{x}) < t_c + 1$, P_i can safely remove $(p, \hat{x})$ according to case (F.2), since $N_{i,p}(\hat{x}) < t_c + 1$ implies that either (i) $(p, \hat{x})$ was maliciously posted, or (ii) a receipt for $(p, \hat{x})$ was not generated.

We conclude that for every honestly posted item (p, x) for which a receipt has been generated, every honest IC peer P_i , will include (p, x) in $B_{i,p}$ at the end of the **Finalization** phase, with $1 - \mathsf{negl}(\kappa)$ probability. $\square$

Given Lemma 3, we prove Theorem 2 below

Proof. For $\theta \geq \Delta + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}$, consider an adversary $\mathcal{A}$ against θ-Confirmable Liveness, that wins $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,0}_{\theta-\mathsf{C.Live}}(1^\kappa, \mathbf{E})$. Assume that conditions (L.1) and (L.2) of $\mathcal{G}^{\mathcal{A},\delta,\Delta,t_c,0}_{\theta-\mathsf{C.Live}}(1^\kappa, \mathbf{E})$ must hold. Namely,

(L.1). For some honest user U, $\mathcal{A}$ provides $\mathcal{C}$ with the message (post, U, x) at global time $\mathsf{Clock} = T$, where T is during a period $p = [T_{\mathsf{begin},p}, T_{\mathsf{end},p}]$, and

(L.2). No **Publishing** protocol execution happens during global time interval $[T, T + \theta]$.

In our BB system, **Publishing** protocol starts at global time $\mathsf{Clock} = T_{\mathsf{end},p}$, which suggests that U engaged at the **Posting** protocol at global time $\mathsf{Clock} = T \leq T_{\mathsf{end},p} - \theta$.

We will show that condition (L.3) can not hold, which implies that $\mathcal{A}$ can not win the game and completes the proof. We analyze the following cases.

(L.3.a) can not hold: By global time $\mathsf{Clock} \leq T + \theta$*, U will obtain a valid receipt* $\mathsf{rec}[x]$ *for x with* $1 - \mathsf{negl}(\kappa)$ *probability.*

Based on the description of the **Posting** protocol in Fig. 6, we show that the waiting time for U is upper bounded by the value $\theta^* = \Delta + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}$. In our computation, we always consider time advancement according to the description of the CS **Posting** protocol and under the restrictions posed in Fig. 3, that is, message delay bound δ and synchronization loss bound Δ. Moreover, recall that $\mathcal{A}$ can not tamper the global clock.

By (L.1), when U broadcasts (x, cr_U), the global time is $\mathsf{Clock} = T$. Hence, taking into account the message delay bound δ, each peer P_i will receive x by global time $\mathsf{Clock}[P_i] \leq T + \delta$.

W.l.o.g., we may assume that the time for verifying the validity of cr_U is the same as verifying a signature (in fact, cr_U could be a signature). Upon receiving x, P_i (i) checks the validity of cr_U (in T_{Vf} time) and (ii) computes and broadcasts the value $(i, (p, x), \mathsf{Sig}(\mathsf{sk}_i, p, x))$ to all other IC peers (in T_{Sig} time) by

$$\mathsf{Clock} \leq T + \delta + T_{\mathsf{Vf}} + T_{\mathsf{Sig}} \ .$$

According to the **Posting** protocol description the honest user U broadcast (x, cr_U) to all peers, so each peer P_i will receive each other honest peers' signatures by global time

$$\mathsf{Clock} \leq (T + \delta + T_{\mathsf{Vf}} + T_{\mathsf{Sig}}) + \delta = T + 2\delta + T_{\mathsf{Vf}} + T_{\mathsf{Sig}} \ .$$

In order for an honest peer P_i to add (p, x) to $B_{i,p}$, it must receive and verify the validity of $N_c - t_c - 1$ signatures from the other peers, as the adversary can also send at most t_c invalid messages on behalf of malicious IC peers[9]. The time required for verification is at most $t_c + (N_c - t_c - 1) \cdot T_{\mathsf{Vf}} = (N_c - 1) \cdot T_{\mathsf{Vf}}$ and the signature share will be created in time T_{ShareSig}. Therefore, each honest P_i will send its TSS share by global time

$$\begin{aligned}\mathsf{Clock} &\leq (T + 2\delta + T_{\mathsf{Vf}} + T_{\mathsf{Sig}}) + (N_c - 1) \cdot T_{\mathsf{Vf}} + T_{\mathsf{ShareSig}} \\ &= T + 2\delta + N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} \ .\end{aligned}$$

The user U will obtain all honest peers' TSS shares by global time

$$\mathsf{Clock} \leq T + 3\delta + N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} \ ,$$

and requires at most $N_c \cdot T_{\mathsf{Vf}}$ to verify all shares (again, $\mathcal{A}$ can send invalid shares). Finally, the user U will require at most T_{Combine} time to combine the TSS shares and obtain her receipt by global time

$$\begin{aligned}\mathsf{Clock} &\leq (T + 3\delta + N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}}) + N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Combine}} = \\ &= T + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}\end{aligned}$$

hence, by internal time

$$\mathsf{Clock}[U] \leq T + \Delta + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}.$$

Consequently, we set the upper bound θ^* for the waiting time of U as

$$\boxed{\theta^* := \Delta + 3\delta + 2N_c \cdot T_{\mathsf{Vf}} + T_{\mathsf{Sig}} + T_{\mathsf{ShareSig}} + T_{\mathsf{Combine}}}$$

[9] The malicious IC peers can send messages arbitrarily. However, for simplicity and without loss of generality for the liveness guarantee, we treat the messages of each malicious peer as a block. Thus, P_i can receive at most t_c malicious messages.

The validity of the receipt that U computes derives directly from the fact that $N_c - t_c > t_s \geq t_c$ and TSS is (t_s, N_c)-EUFCMA-secure with $1 - \mathsf{negl}(\kappa)$ probability.

(L.3.b) can not hold: There is a moment T' s.t. for every $T'' \geq T'$, (p, x) is included in the view of AB.

To prove this statement, we apply Lemma 3, which states that at the end of the **Finalization** phase of the **Publishing** protocol, every honest IC peer P_i will include (p, x) in $B_{i,p}$ at the end of the **Finalization** phase, with $1 - \mathsf{negl}(\kappa)$ probability. Recall that Lemma 3 holds under the partial synchronicity assumptions considered in the theorem's statement.

Therefore, by Lemma 3, we get that at the **Publication** phase, every honest peer P_i will threshold sign and send to AB the message $\big((p, B_{i,p}), \mathsf{ShareSig}(\mathsf{tsk}_i, (p, B_{i,p}))\big)$ that includes the TSS share $\mathsf{ShareSig}(\mathsf{tsk}_i, (p, x))$ for (p, x). Consequently, AB will receive $N_c - t_c$ valid TSS shares on (p, x) by the end of period p, so it will (i) add (p, x) to B_p, (ii) compute a TSS signature on (p, x) as $\mathsf{TSign}(\mathsf{tsk}, (p, x)) \leftarrow \mathsf{Combine}\big(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{N_c}, (p, x), (j, \sigma_j)_{j \in S}\big)$ and (iii) include $\mathsf{TSign}(\mathsf{tsk}, (p, x))$ in the published record $\mathsf{ABreceipt}[p, B_p] := \big((p, B_p), \mathsf{TSign}(\mathsf{tsk}, (p, B_p))\big)$.

We conclude that there is a moment $T' > T_{\mathsf{end},p}$ s.t. for every $T'' \geq T'$, (p, x) is included in the view of AB, defined as the union of the agreed and published BB records preceding the period p' that T' belongs (therefore also p).

□